MacTech Magazine
Vol. 17, No. 09 • September 2001

MacTech®

The Journal of Macintosh Technology and Development

QuickTime

Video Effects

FX

Using Video Effects in QuickTime Movies

By Tim Monroe

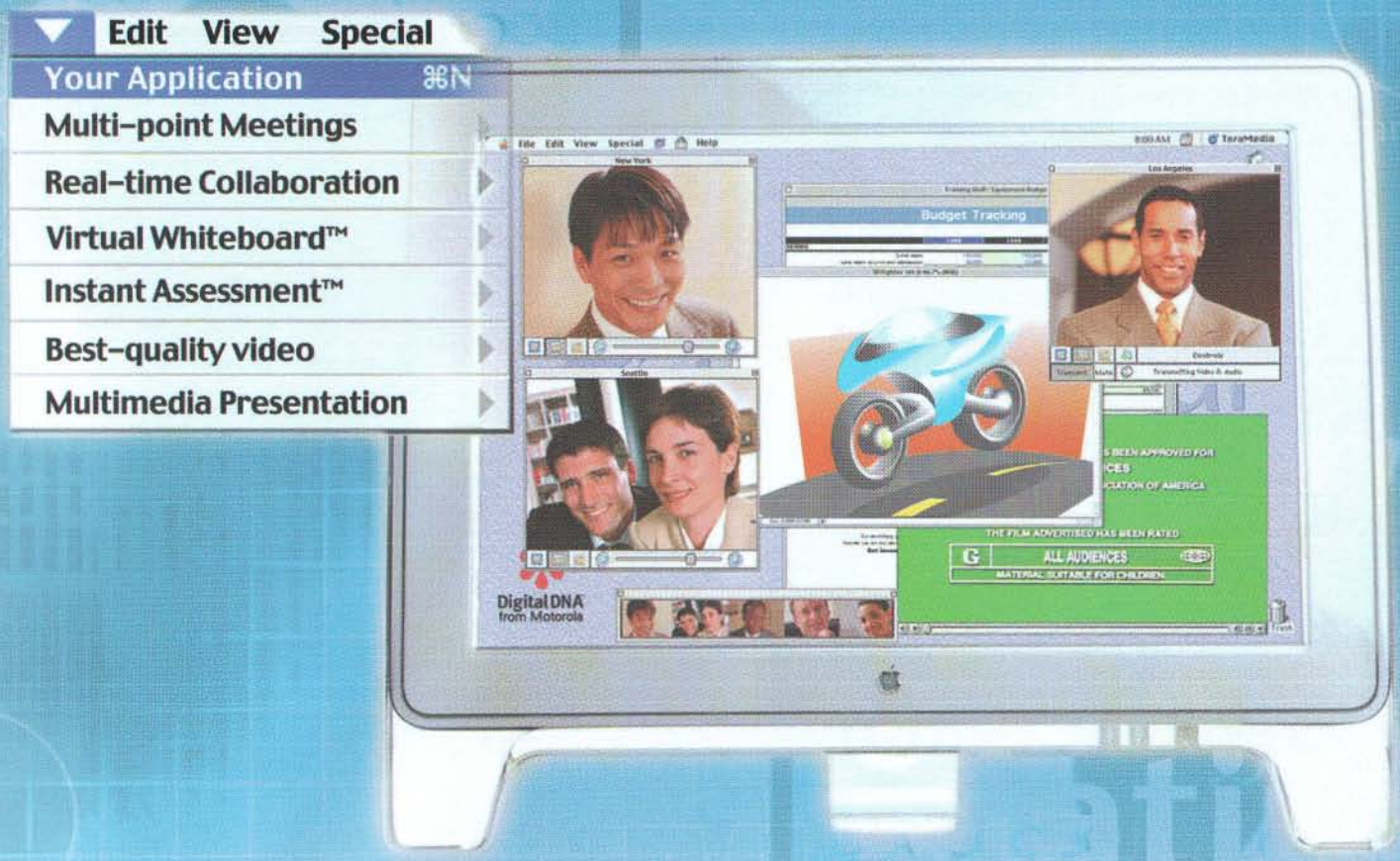
$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.

09
0 32128 74887 8

LA to New York

(and Seattle and Miami)

in 150 milliseconds or less*



Call for a TeraMedia® demo

TeraGlobal COMMUNICATIONS

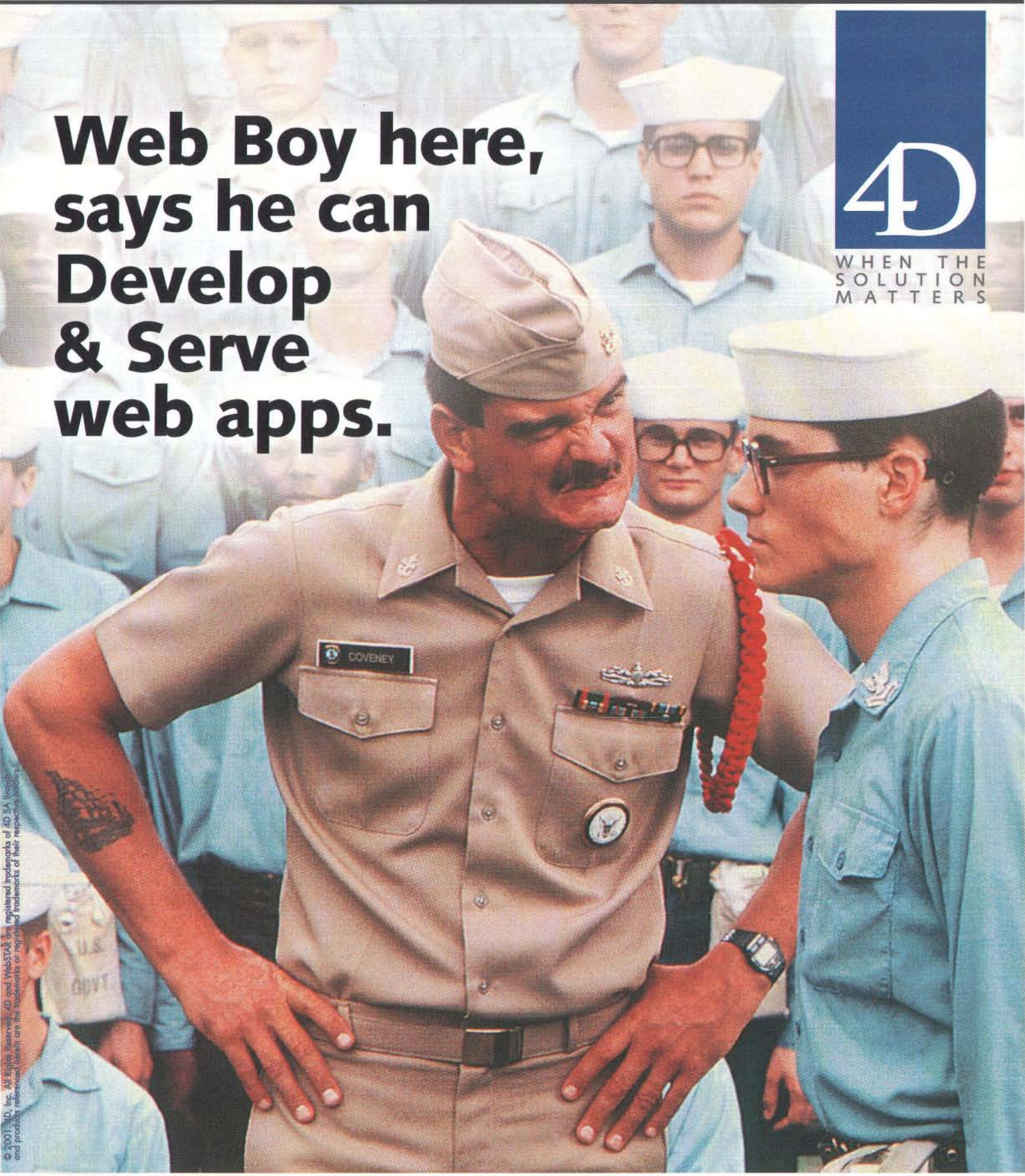
(858) 404.5500 ext 5660
www.teraglobal.com
sales@teraglobal.com

Digital DNA from Motorola

Interact. Collaborate. It's highest-quality multi-point video conferencing **and** true real-time multimedia collaboration. It's all done in software. Any number of users. Over any distance. At any time. In real time. It flies, you don't.

* Actual video latency varies with bandwidth. In most cases, TeraMedia® latency is below the threshold of human perception.

TeraMedia® by TeraGlobal



Web Boy here, says he can Develop & Serve web apps.

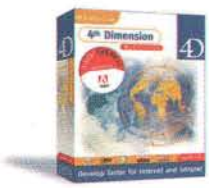
4D

WHEN THE SOLUTION MATTERS

INTRODUCING

4D 6.7 Web Edition $449

BUNDLED FOR A LIMITED TIME WITH ADOBE® GOLIVE® 5.0



Professional web application development from design to finish.

1.800.881.3466

www.4D.com/webedition

MacTech®
The Journal of Macintosh Technology & Development

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@mactech.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

Contents

September 2001 • Volume 17, Issue 09

Feature Articles

Mac OS X

38 ***Internationalizing Cocoa Applications - A Primer for Developers and End Localizers***

by Andrew C. Stone

Columns

Beginning WebObjects 5

14 **Part 1 - Presenting WebObjects 5**

by Emmanuel Proulx

Advanced WebObjects 5

30 **Deep Into the Request/Response Loop**

by Emmanuel Proulx

Programmer's Challenge

68 **Nassi-Schneiderman**

by Bob Boonstra

Network Management

92 **A Network Geek in the Big Apple**

by John Welch

Mac OS X . . . . . Page 38

FX . . . . . Page 44

Cover Story

44 **FX**

By Tim Monroe

Programming Techniques

6 **The Under-Used UserPane**

Presenting a Whole Slew of Nifty Controls

by Spec Bowers

Programming Philosophy

22 **Mac vs. Unix Traditions**

How to combine the best of both, with Mac OS X, without gathering the worst

by Marcelo Amarante Ferreira Gomes

The world is a very different place since the last time you saw a MacTech. It's different for all of us... Americans, Europeans, Asians... kids, grandparents, adults... those who are religious, and those who are not. The world is very different for everyone, everywhere.

Now is the time for us to take the tragedy of September 11th, and work relentlessly to create whatever good we can from this evil. In part, this means that we use this juncture in mankind's history to take world unification to the next level — and learn to live and work better with one another.

In addition, many of the leaders around the world are urging that people return to their level of efforts before the attack. They say this because one of the best things that free people can do in the face of terrorism is not let it win, slow us down, or stop us. This means driving forward with our projects, our plans, our industry and our economy. I strongly encourage you to do the same... as we are here.

While stated millions of times at this point... and cannot be understated... our hearts, thoughts and prayers go out to those who lost their lives, or loved ones on that horrific day.

There have been many "moments of silence" as of late. We're going to take a moment now, and do the "print" version of this. The balance of this page will remain blank in memory of those that lost their lives, those who selflessly leapt at the chance to save others, and for all those grieving their losses.

Neil Ticktin, Publisher

Looking for more speed, scalability and reliability from your database? c-tree Plus® is the perfect fit!

What shape is your database project? A tiny embedded database that requires a small footprint but demands rigorous functionality? Or a huge multi-platform, multi-user, multi-headache project? Have you struggled to find a better solution for your data access requirements, only to feel like you're sacrificing one requirement over another?

FairCom has delivered uncompromising database technology to commercial developers for over twenty years, with the primary goal of keeping control in the hands of the developer. Our customers rely on the speed, flexibility, scalability and reliability of c-tree Plus. The high performance and low cost of ownership make c-tree Plus an excellent choice for all sizes of database development projects.

Proven Database Technology

Besides the ISAM-level control and speed, perhaps the most important reason why small development houses and Fortune 1000 companies have chosen FairCom technology is the superior service offered by our sales and support staff. Our development staff understands the challenges you face, and we'll put our twenty years of experience to work to help you build better solutions.

Single Solution for Diverse Implementation

- Vertical Markets
- Embedded Systems
- Web & ASP Markets
- E-Commerce
- Smart-Cards
- Web-Enabled Appliances

Comprehensive Feature Set

- Royalty-free single user and multi-user support with full source code
- Client/server and custom server support
- Robust server side SDK - build your own application specific database server with full source available!
- Thread compliant + portable thread API
- Full featured transaction processing with savepoints, abort and full rollback
- Comprehensive security and encryption features
- Small footprint
- Fixed and variable length records and keys
- Store any data type - up to 18 million terabytes!
- Dynamic space reclamation
- ODBC and Crystal Reports™ drivers
- Full ISAM functionality
- Powerful stand-alone index support

Multiple-Platform Support

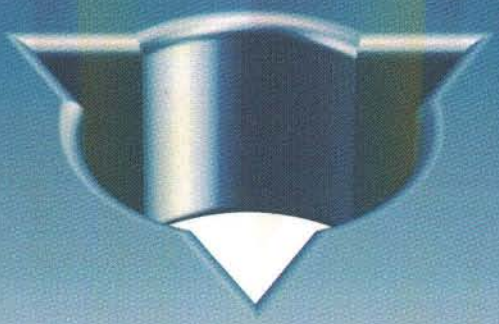
Mac OS, Mac OSX, MkLinux, Linux (PPC,Intel, SPARC, Alpha), Windows 95/98/ME/2000/NT, Novell Netware, Solaris (SPARC, Intel), Sun OS, OS/2, AIX, HP UX, SCO UnixWare, Interactive, AT&T Sys V, QNX, 88OPEN, FreeBSD, Lynx, Banyan Vines, and more...

Supports ADSP, SPX, TCP/IP

FairCom®

QUALITY•THROUGH•EXPERIENCE

www.faircom.com/maca



FairCom Offices

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

Mac Support Since 1985 • USA. 800.234.8180 • info@faircom.com

By Spec Bowers

The Under-used UserPane

Presenting a Whole Slew of Nifty Controls

Introduction

Could you use a QuickTime control, an MLTE control, an HTML Renderer control, a StoneTable control, a WASTE control, a Color Picker control? They're all here – made out of a UserPane.

Of all the controls Apple has created in recent years, the most flexible is the UserPane. It is also the hardest to use. Most of us probably just write special–purpose code in our update handlers or mouse–down handlers rather than bother with the intricacies of a UserPane.

With a simple wrapper class, the UserPane is very easy to use – and it makes other packages easier to use. We have made UserPane controls for QuickTime, MLTE, HTML Rendering, the Color Picker, the StoneTable list manager replacement, and the WASTE text engine. These packages are easier to use thanks to the UserPane.

Figure 1. *Four kinds of UserPane*

Wrapping a UserPane control around QuickTime, MLTE, or other packages makes the code easy to reuse and greatly simplifies your event handling code. The QuickTime control is almost as simple as a standard pushbutton; the MLTE control is easier than TextEdit.

Your event loop already has code for HandleControlClick, HandleControlKey, SetKeyboardFocus, Activate/DeactivateControl, Hide/ShowControl, and IdleControls. When you wrap a UserPane control around a package like QuickTime, it automatically responds to your existing control handling code. You can put controls for QuickTime, MLTE, HTML Rendering, etc. in a window, a dialog, inside a tab panel – anywhere you put a standard control – and it just works.

This article:

- describes the functions of a UserPane;
- presents a C++ wrapper class that makes it easy to use a UserPane;
- presents several examples of UserPanes;
- shows how to handle a UserPane in a window or dialog

UserPane Functions

Apple defines several callback functions for a UserPane control:

- A DrawProc draws the content of your control. It might be called to draw a part of the control but usually draws the entire control.
- A HitTestProc returns the part code of the control where the mouse–down occurred. We usually just detect a mouse–down anywhere in the control and return a partcode which represents the entire control.
- A TrackingProc tracks a control while the user holds down the mouse button.
- An IdleProc performs idle processing.
- A KeyDownProc handles keyboard events.
- An ActivateProc handles activate and deactivate events.
- A FocusProc handles keyboard focus – e.g. for Set/AdvanceKeyboardFocus.
- A BackgroundProc sets the background color or pattern for embedded controls.

Spec Bowers is the founder, cook, and chief bottle washer at Bowers Development. He has been developing programming tools for most of his career. You can contact him at bowersdev@aol.com or see the web page at http://members.aol.com/bowersdev

To install a callback you first create a UPP, then pass it to the control via SetControlData. Later, to prevent a memory leak, you should dispose the UPP. Alternatively, you can adopt a "create once and reuse" protocol. Whatever way you do it, installing a callback is a nuisance. The AMUserPane makes it very simple to install a callback.

The AMUserPane Class

The AMUserPane class provides functions for easily installing callbacks and for disposing of UPPs when the control is discarded, provides some standard callback functions, and provides some utility functions to simplify using a UserPane.

To install a DrawProc, just call "SetDrawProc ()":

```
//———
void AMUserPane::SetDrawProc ()
{
  mDrawUPP = NewControlUserPaneDrawUPP (StaticDrawProc);
  ::SetControlData (mControl,
            kControlNoPart,
            kControlUserPaneDrawProcTag,
            sizeof (mDrawUPP),
            (Ptr)&mDrawUPP);
}
```

There are similar functions for installing a TrackingProc, KeyDownProc, ActivateProc, etc.

AMUserPane declares data members for each callback function:

```
  ControlUserPaneDrawUPP        mDrawUPP;
  ControlUserPaneHitTestUPP     mHitTestUPP;
  ControlUserPaneTrackingUPP    mTrackingUPP;
  ControlUserPaneIdleUPP        mIdleUPP;
  ControlUserPaneKeyDownUPP     mKeyDownUPP;
  ControlUserPaneActivateUPP    mActivateUPP;
  ControlUserPaneFocusUPP         mFocusUPP;
  ControlUserPaneBackgroundUPP    mBackgroundUPP;
```

Its constructor initializes each UPP to nil, and its destructor disposes each (non-nil) UPP:

```
//———
AMUserPane::AMUserPane ()
{
  mDrawUPP = nil;
  mHitTestUPP = nil;
  mTrackingUPP = nil;
  mIdleUPP = nil;
  mKeyDownUPP = nil;
  mActivateUPP = nil;
  mFocusUPP = nil;
  mBackgroundUPP = nil;
}

//———
AMUserPane::~AMUserPane ()
{
  if (mDrawUPP != nil) {
    DisposeControlUserPaneDrawUPP (mDrawUPP);
  }
  if (mHitTestUPP != nil) {
    DisposeControlUserPaneHitTestUPP (mHitTestUPP);
  }
  if (mTrackingUPP != nil) {
    DisposeControlUserPaneTrackingUPP (mTrackingUPP);
  }
  if (mIdleUPP != nil) {
    DisposeControlUserPaneIdleUPP (mIdleUPP);
  }
  if (mKeyDownUPP != nil) {
    DisposeControlUserPaneKeyDownUPP (mKeyDownUPP);
  }
  if (mActivateUPP != nil) {
    DisposeControlUserPaneActivateUPP (mActivateUPP);
  }
  if (mFocusUPP != nil) {
    DisposeControlUserPaneFocusUPP (mFocusUPP);
  }
  if (mBackgroundUPP != nil) {
    DisposeControlUserPaneBackgroundUPP (mBackgroundUPP);
  }
}
```

Look back at SetDrawProc and you'll see that it installs a callback to a function named "StaticDrawProc". AMUserPane provides a member function, DoDraw, which is overridden in each subclass. The StaticDrawProc is a glue function which dispatches to the particular DoDraw of the subclass – the QuickTime DoDraw, or the MLTE DoDraw, for example. During initialization we store a pointer to a specific instance of AMUserPane in the control's RefCon. In each callback we retrieve the control's RefCon, cast it to an AMUserPane pointer, then call the member function.

```
//———
void AMUserPane::Initialize (
  ControlHandle   inControl)
{
  mControl = inControl;
  ::SetControlReference (mControl, (SInt32)this);
}

//———
pascal void   AMUserPane::StaticDrawProc (
  ControlHandle control,
  SInt16          part)
{
  AMUserPane*   pane = (AMUserPane*)::GetControlReference
(control);

  pane->DoDraw (part);
}

//———
void AMUserPane::DoDraw (
  SInt16   part)
{
  // override in each subclass
}
```

AMUserPane is a base class; it provides common code for a wide variety of UserPane controls. We have made half a dozen subclasses. Let's take a look at some of them.

A ColorSwatch UserPane

Our simplest UserPane is a wrapper around the Color Picker. It paints the control's rectangle with a color. If the user clicks the UserPane, it invokes the Color Picker, then redraws the rectangle with the selected color. We overrode DoDraw and DoTracking. The Initialize function calls SetDrawProc and SetTrackingProc to install callbacks.

```
//———
void AMColorSwatch::DoDraw (
  SInt16      /* part */ )
{
  RGBColor    saveColor;
  Rect        rect;

  ::GetForeColor (&saveColor);
  ::RGBForeColor (&mSwatchColor);
  GetControlRect (&rect);
  ::PaintRect (&rect);
  ::RGBForeColor (&saveColor);
}

//———
ControlPartCode    AMColorSwatch::DoTracking (
  Point         startPt,
  ControlActionUPPactionProc)
{
  ControlPartCode    result = 0;
  Point         dialogPos = {0, 0};
  Str255        prompt = "\p";
  RGBColor      outColor;

  if (::GetColor (dialogPos, prompt, &mSwatchColor,
&outColor)) {
    mSwatchColor = outColor;
    DoDraw (0);
    result = 99;  // any non-zero code
  }

  return result;
}

//———
void AMColorSwatch::Initialize (
  ControlHandle   inControl)
{
  AMUserPane::Initialize (inControl);

  SetDrawProc ();
  SetTrackingProc ();
}
```

An HTML Pane – Glitches and Solutions

The HTML pane is only slightly more complex but illustrates two glitches. The first time we put it inside a Tab control it worked pretty well. Clicking a tab results in HideControl/ShowControl of the panels. Each panel is a simple UserPane with embedded controls. When we hide or show a panel, the Control Manager hides or shows any embedded controls, including our custom UserPane controls. (This by the way is one of the advantages of turning QuickTime, MLTE, etc. into UserPane controls – HideControl, ShowControl and other Control Manager functions work the same as with standard controls.)

There was a glitch, though. When we deactivated the window, suddenly one of our hidden UserPane controls drew something. The Control Manager doesn't call the DrawProc of a hidden control but it may call the ActivateProc. We added a simple "IsVisible" call to test for visibility inside any of our callback functions that might draw anything.

```
//———
void AMHTMLPane::DoActivate (
  Boolean     activating)
{
  Rect      cntlRect;

  if (IsVisible ()) {
    if (activating) {
      HRActivate (mHTMLRec);
    } else {
      HRDeactivate (mHTMLRec);
    }
  }
}
```

This didn't completely solve the problem, however. We saw the HTMLPane's scrollbars even when the pane was hidden. This was because the scrollbars were not properly embedded within the HTML UserPane. The HTML Rendering Lib creates scrollbars on the fly as needed. Those scrollbars end up embedded in the root control. Because they are not embedded in the HTML UserPane they are not hidden/shown along with the pane. Our solution is to look for newly created controls in the root control and embed them in our UserPane control.

Without this piece, you're not done.

It's not enough just to write solid code anymore. Users expect an easy-to-use installer when they buy your software.

InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.

Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers. Even create demoware in minutes.

Visit **http://www.stuffit.com/installermaker/** to also learn about creating Mac® OS X compatible installers with InstallerMaker.

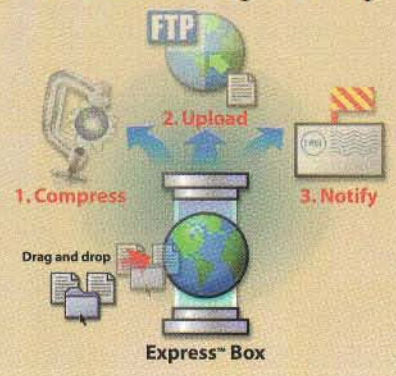
Then you're done. **StuffIt InstallerMaker**

The complete installation and distribution solution.™

Get your beta testing started faster.

Guarantee your builds arrive safe and sound! New **StuffIt Express**™ automates file compression and transfer tasks to eliminate costly errors every time you exchange data. By creating custom drop box applications, you and those you communicate with can accomplish complex file transfer tasks with a simple drag and drop. Choose from up to 26 file transfer steps with no coding required. Boost productivity and save time by distributing your custom Express boxes to everyone on your list.

http://www.stuffit.com/express/enterprisemac.



Before we call any function that might create other controls we call CountRootControls. Afterwards, we call EmbedNewControls. If there are more controls afterwards than before, those new controls should be embedded in our UserPane, not in the root control.

```
//———
UInt16   AMUserPane::CountRootControls ()
{
  UInt16        numControls = 0;
  WindowRef   theWindow;
  ControlRef    root;

  theWindow = GetOwnerWindow ();
  ::GetRootControl (theWindow, &root);
  ::CountSubControls (root, &numControls);

  return numControls;
}

//———
void AMUserPane::EmbedNewControls (
  UInt16      inBeforeNum)
{
  WindowRef   theWindow;
  ControlRef    root;
  UInt16        afterNum;
  UInt16        i;
  ControlRef    child;

  theWindow = GetOwnerWindow ();
  ::GetRootControl (theWindow, &root);
  ::CountSubControls (root, &afterNum);
  for (i = afterNum; i > inBeforeNum; i--) {
    ::GetIndexedSubControl (root, i, &child);
    ::EmbedControl (child, mControl);
  }
}
```

In most of our UserPanes' Initialize functions we call CountNewControls and EmbedNewControls just in case subcontrols are created. The HTMLPane's Initialize is typical. We get a "before" count of root controls, create a new HRReference, then set its bounds to the UserPane control's bounds. We install a DrawProc, TrackingProc, and ActivateProc. Finally, we embed the newly created controls (scrollbars) in the UserPane.

```
//———
void AMHTMLPane::Initialize (
  ControlHandle    inControl)
{
  AMUserPane::Initialize (inControl);

  OSErr         err = noErr;
  UInt16        beforeNum;
  Rect          cntlRect;
  GrafPtr   ownerPort;

  beforeNum = CountRootControls ();

  GetControlRect (&cntlRect);

  ownerPort = (GrafPtr) GetWindowPort (GetOwnerWindow ());
  err = HRNewReference (&mHTMLRec, kHRRendererHTML32Type,
ownerPort);

  if (err == noErr) {
    HRSetRenderingRect (mHTMLRec, &cntlRect);
    HRSetDrawBorder (mHTMLRec, true);

    SetDrawProc ();
    SetTrackingProc ();
    SetActivateProc ();
  }

  EmbedNewControls (beforeNum);
}
```

The HTMLPane was now working well – until we viewed an HTML file that had a frameset. Suddenly, there were two new scrollbars and they were not embedded properly. So we added CountRootControls and EmbedNewControls to the DoDraw function. Other than that, the DoDraw is very simple.

```
//———
void AMHTMLPane::DoDraw (
  SInt16      part)
{
  UInt16      beforeNum;
  Rect        cntlRect;

  beforeNum = CountRootControls ();

  GetControlRect (&cntlRect);
  HRSetRenderingRect (mHTMLRec, &cntlRect);

  RectRgn (mHTMLRgn, &cntlRect),
  HRDraw (mHTMLRec, mHTMLRgn);

  EmbedNewControls (beforeNum);
    // in case last click created new scroll bars
}
```

The DoTracking function is very simple. About all it does is ask the HTML Rendering library to handle the event. We could have synthesized a mouse-down event from the Start Point but instead we call an external function to get the current event record from our main event-handling code.

```
//———
ControlPartCode    AMHTMLPane::DoTracking (
  Point         startPt,
  ControlActionUPP actionProc)
{
  WindowRef   owner = GetOwnerWindow ();

  ::SetPortWindowPort (owner);
  ::HRIsHREvent (GetCurrentEventRecord ());

  return 99;  // any non-zero code
}
```

Using a UserPane

Okay, so we have written a UserPane class. How do we use it? That's the easy part. First, create a UserPane control resource. For a window or a dialog, create a CNTL with procID 256 and initial value 318. This value turns on feature bits for SupportsEmbedding, SupportsFocus, WantsIdle, WantsActivate, HandlesTracking, and GetsFocusOnClick. For a dialog, in its DITL resource create an item of type Control and set its ID to the resource ID of the CNTL resource.

Wherever you declare the variables (data members) for your window or dialog, declare an instance of the UserPane class. We find it convenient also to declare a ControlHandle. You'll also have to #include the UserPane class's header.

```
#include "AMColorSwatch.h"
  ControlHandle  mSwatchHandle;
  AMColorSwatch  mSwatchPane;
```

When you create a window, get a ControlHandle to the UserPane control, then initialize the instance of the UserPane class.

```
  mSwatchHandle = ::GetNewControl (CNTL_Swatch, window);
  mSwatchPane.Initialize (mSwatchHandle);
```

In your window's event handling code for a mouse-down, call the usual FindControl. If the click is in the UserPane control, call the usual TrackControl or HandleControlClick.

```
  if (whichControl == mSwatchHandle) {
```

A New System Has Arrived.

Don't Get Caught With Your Mac Down!

No Mac is complete without Timbuktu Pro, the premier remote control and file transfer software for Mac OS for over ten years.

No network is complete without the smart systems management of netOctopus.

Both tools are newly rewritten for OS X, bringing the power and simplicity of Netopia software to the world's most advanced operating system. For deployment, training and support of all your Macs (and PC's!) Timbuktu and netOctopus are indispensable.

Are you ready to migrate? We're ready to take you there.

macosxready.com

Timbuktu Pro® netopia® netOctopus

```
    if (HandleControlClick (mSwatchHandle, where,
curEvent.modifiers, nil) != 0) {
    }
  } else if (mSwatchPane.ClickSubControl (whichControl,
where)) {
    // ClickedSwatch ();
  }
```

What is "ClickSubControl"? If the UserPane has any subcontrols, e.g. scrollbars, then FindControl may find that the click was in the scrollbar. ClickSubControl checks to see if the click was in one of the UserPane's subcontrols. If it was, then ClickSubControl passes the click along to the UserPane class's DoTracking method and returns true. If the click was not in a subcontrol, then ClickSubControl returns false.

```
//———
Boolean     AMUserPane::ClickSubControl (
  ControlRef    inControl,
  Point         inWhere)
{
  UInt16        numSubs;
  UInt16        i;
  ControlRef    sub;

  ::CountSubControls (mControl, &numSubs);
  for (i = 1; i <= numSubs; i++) {
    ::GetIndexedSubControl (mControl, i, &sub);
    if (inControl == sub) {
      DoTracking (inWhere, nil);
        // treat it as a click in the userPane
      return true;
    }
  }
  return false;
}
```


The example application doesn't have any UserPanes in a dialog but it's easy to do. After creating the dialog, call GetDialogItemAsControl to get a ControlHandle to the UserPane. Then Initialize the UserPane instance with the control handle. The Control Manager and Dialog Manager will take care of almost everything. You just add a case to your dialog's switch statement.

There is one other piece of code you will have to add if your UserPane has subcontrols. You'll have to add a Filter function because the Dialog Manager doesn't know anything about the subcontrols. The Filter function will have code like this:

```
if (mQuickTimePane.FilterSubControls (ioEvent)) {
  *outItemHit = kQuickTimePane;
  return true;
}
```

FilterSubControls checks to see if the event is a mouse-down. If it is, then FilterSubControls calls FindWindow and FindControl, then calls ClickSubControl, which we saw earlier.

```
//———
// if the event is for one of our subcontrols
// then process it as if it were for us;
// pass back true to tell Dialog Manager
// that event has been processed
//
Boolean   AMUserPane::FilterSubControls (
  EventRecord    *ioEvent)
{
  Boolean      filtered = false;
  UInt16       numControls = 0;
  WindowPtr    whichWindow;
  ControlHandle whichControl;
  Point        localWhere;
  short        partCode;
  UInt16       i;
  ControlRef     sub;

  ::CountSubControls (mControl, &numControls);
  if (numControls > 0) {
    if ((ioEvent->what == mouseDown)
    && (FindWindow (ioEvent->where, &whichWindow) == inContent))
{
      SetPortWindowPort (whichWindow);
      localWhere = ioEvent->where;
      GlobalToLocal (&localWhere);
      FindControl (localWhere, whichWindow, &whichControl);
      if (ClickSubControl (whichControl, localWhere)) {
        return true;
          // => click was for this userPane
      }
    }
  }
  return filtered;
}
```

Summary

We've described two of our UserPanes – the Color Picker and the HTML Renderer. The other four UserPanes – for QuickTime, MLTE, StoneTable, and WASTE – are similar. The AMUserPane class, all six UserPane classes, and the example application are available as source code. The controls are useful and easy to use. If you use any of them we would be interested in hearing from you.

A Plug

These UserPane classes are part of the AppMaker library. The example was created using AppMaker, which generated both the resources and the source code. Whether you use AppMaker or you do it by hand, I think you will find that these UserPanes are very useful widgets to have in your toolbox. MT

My files are important.

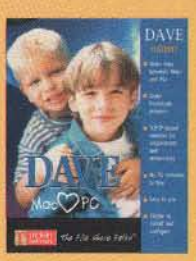
Very important.

I only share my files with those I trust.

I trust DAVE®.

Mac to PC, PC to Mac. Cross-platform file and print sharing is too vital to your business to risk. Trust Thursby, the company with 15 years experience. Trust DAVE, the solution with a proven track record. Share files and printers across a network with no barriers. DAVE installs on your Mac with no additional software required for the PC. It's fast, secure and easy to use. Download a free evaluation today!

Trust DAVE®.



Macworld 1/2

www.thursby.com/tech

THURSBY Software

The File Share Folks™

Built for Mac OS X

By Emmanuel Proulx

Part 1 - Presenting WebObjects 5

Introduction & Installation

Preface

This new column deals with WebObjects 5. It is meant to be easy and fun to read. This column talks about developing a Web application using WebObjects 5, in Java. Originally, this column was a book, with hundreds of pages. It has been cut down a lot to accommodate space restrictions.

Those interested in learning to build Web applications using WebObjects 5 should read this column. The only prerequisite is being Java programmers of at least intermediate level.

Other knowledge is not necessary, although understanding HTML basics can help tremendously because WebObjects borrows many characteristics from this language. If you don't know HTML at all but you still want to learn WebObjects, just skip the paragraphs that talk about HTML. For your convenience, these are usually marked with the symbol <HTML>. You won't get into any trouble, but you will not understand how WebObjects works behind the scene.

The same thing goes with databases. While knowing databases and SQL isn't necessary, you have to have basic knowledge if you're going to read the sections about databases. Database software is necessary if you're going to try out WebObjects' database features. I tried to keep all of the database interfacing and programming contents in separate sections for convenience.

Note that someone who masters Java programming, client/server design, HTML and Web Design as well as databases and SQL will be able to learn WebObjects radically faster.

What Is WebObjects?

I started this new job and the job description was "Java Developer". I was expecting to use one of these Java IDEs like Visual Café or Visual Age. My new boss said, "No, here we use WebObjects". I had heard about that tool, but I didn't know it was a Java IDE. It wasn't.

WebObjects is an environment for developing interactive Web Sites. The Java language is being used to implement Server-side behavior; the client side is mostly HTML, not necessarily Java Applets or even JavaScript for that matter.

WebObjects is much more than that. If also falls into the Application Server category. What is that? In the past, a "Server" would consist of just a Database. Nowadays a Server can also handle distributed transactions, multi-tier architecture, load-balancing, business logic, Enterprise JavaBeans, and other concepts and technologies involving more advanced server-side intelligence. An environment that helps with the development and deployment of such complex Servers is called an Application Server.

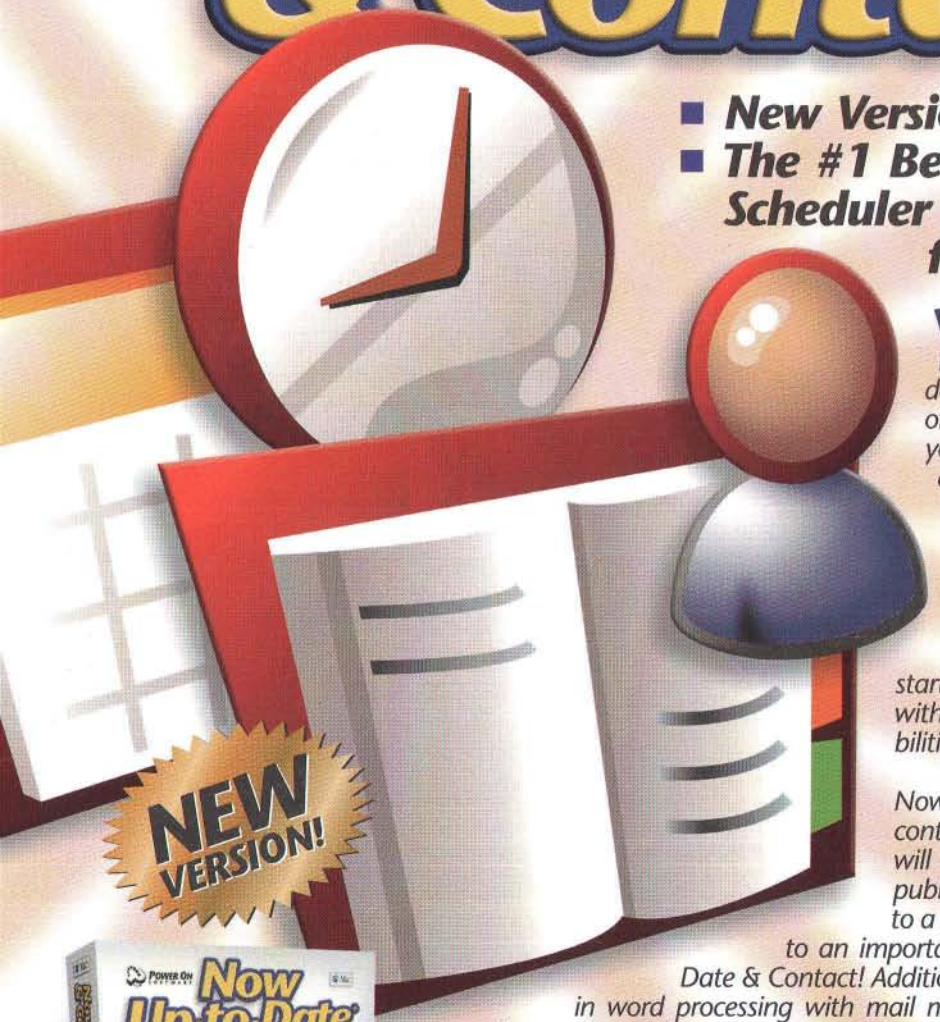
WO consists of multiple tools that are tightly integrated.

- **Project Builder:** lets you browse the source code and manage the project.
- **WebObjects Builder:** lets you assemble a Web Page.
- **Enterprise Object Modeler:** lets you connect your Web Application to a database.
- **Interface Builder:** lets you create Java applications and applets that connect to the WebObjects server for executing business logic and database access.
- **Direct To Web:** a wizard that creates a complete database-driven customizable Web application.
- **Direct To Java Client:** a wizard that creates a complete database-driven customizable Java program or applet.
- **Monitor:** a remote task monitoring and performance analysis utility.

Other smaller utilities like a 'Diff' utility, debugging tools, etc.

Emmanuel Proulx is a Course Writer, Author and Web Developer, working in the domain of Java Application Servers. He can be reached at emmanuelp@theglobe.com.

Now Up-to-Date® & Contact®



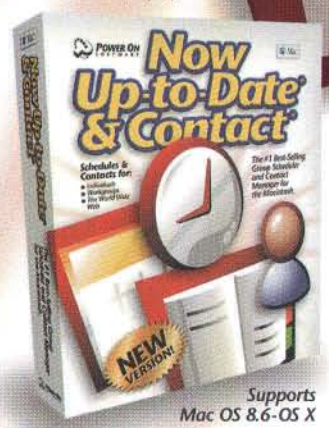
- **New Version!**
- **Mac OS X!**
- **The #1 Best-Selling Group Scheduler and Contact Manger for the Macintosh!**

You're a busy person. You've got things to do, people to see, and a seemingly infinite list of things to keep track of. Whether it's just yourself or a department at a Fortune® 500 company, you (and everyone that works with you) need to be organized. Maybe you've tried other ways: paper planners, lots of sticky notes, other software. But now you can't afford to waste any more time or effort. You need a serious solution. One that's so easy to use that you can get started right out of the box. One that will grow with you. One that has the features and capabilities you need.

Now Up-to-Date & Contact is the internet-savvy contact manager. With a click of the mouse, you will be soaring through cyberspace. Want to publish your calendar on the web, or get a map to a meeting, or even find the restaurant nearest to an important client? It's all a snap with Now Up-to-Date & Contact! Additional features you'll rave about include: built-in word processing with mail merge, automated fax creation, integration with your favorite email software, advanced Palm synchronization, and the amazing new Grab-'n-Go™ that allows you to instantly grab information from virtually any source and have Now Up-to-Date & Contact create any necessary follow-up reminders.

With its amazing power and phenomenal speed, combined with unparalleled ease of use, it's no wonder that Now Up-to-Date & Contact has won the coveted Eddy Award for best Information Manager, seven World Class Awards, and a host of rave reviews.

Get Now Up-to-Date & Contact at www.poweronsoftware.com or, at a retailer near you.



Supports Mac OS 8.6-OS X

MacHome · MacToday · Macworld · Macworld's Home Choice · MBC · Eddy

POWER ON SOFTWARE

@DEPOT

Built for OS X & Classic OS

Fully Mac OS X & Palm Compatible.

Phone: 800-344-9160 ■ 614-413-4000 Fax: 614-413-4100 ■ www.poweronsoftware.com

Other modules are included, yet they are not visible:

- **Web Server Adaptor:** plugs into your Web Server to connect it to WebObjects.
- **JDBC Database Adaptor:** connects WebObjects to a database driver.
- **The frameworks**, which are powerful programming libraries. They support the whole WebObjects system. We refer to them as the Foundation, WebObjects and Enterprise Object Frameworks. The preceding tools make use of the Frameworks, and even generate some code that uses them.

How do these tools work with each other? The **Figure 1** illustrates their interactions.

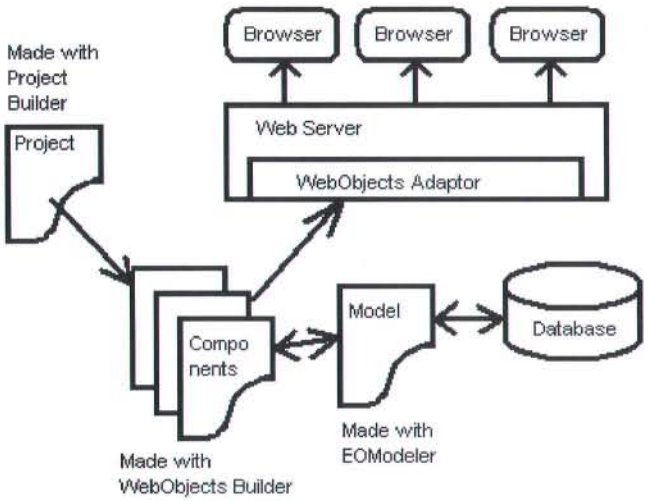


Figure 1. Interactions between the tools

In a typical WebObjects application, the focus is the project file, which points to the different components of the application. The Project Builder manages this file. Then, to create the Web Components (Web pages), you usually run the WebObjects Builder program. When a component makes use of a database, you have to use a model to link the tables of the database to your system's objects. This model is created using the EO Modeler. To let a user connect to your site, you need a Web Server containing the WebObjects Adaptor. There's more on each of these parts later on.

Lastly, WebObjects also is an IDE for developing Cocoa and Carbon applications, but this column deals only with Web applications.

Installation Information

First you must know that there are two distinct packages that can be installed with WebObjects 5. Each has its own specific use and list of installed components.

- **Developer edition:** Contains all the graphical tools and libraries. Used for the development environment. At the time this was written, only the Mac OS X would support the Developer Edition.

- **Deployment edition:** contains only the deployment tools and the run-time libraries. Used for the deployment environment. Can be installed on top of the Developer edition on Mac OS X. The Deployment Edition is available for a variety of platforms.

This column generally talks about the Developer edition. The Deployment edition will be covered in a separate article.

To develop with WebObjects Developer Edition, you need a Mac OS X environment with at least 128 megabytes of RAM and 600 megabytes of hard disk space. That said, remember this golden rule: you never have enough RAM and hard disk space!

Once you're done with your development, you can deploy your application on either Mac OS X, Windows 2000, or Solaris.

You will also need a supported Web server. On the Mac OS X, the default server is Apache, and works great.

If speed doesn't matter that much to you, you can get any Web Server with CGI capability. Else, you have to get a Web Server compatible with either the NSAPI or WAI standards (Netscape's Web Servers), ISAPI (Microsoft's Web Servers) or Apache. Of course, select a Web Server that works on your deployment platform.

If you need to access a database, you want to use one that has a JDBC 2.0 driver. WebObjects comes bundled with an evaluation version of OpenBase. But on deployment platforms you may want to go for an industrial-strength database, like Oracle, Informix, Sybase, DB/2, etc. You can also connect to any ODBC-compliant database on Windows using the ODBC/JDBC bridge, but that would make things slower.

NOTE: Not many database products are available on Mac OS X, so you may want to purchase a complete version of OpenBase (www.openbase.com).

Installation

Pre-Installation Steps

It is very important to install your Web Server before you install WebObjects.

NOTE: if you already installed WebObjects and you want to install a Web Server afterward, the only issue is with the "cgi-bin" folder. WebObjects has already copied its required executables in a temporary "cgi-bin" folder. After setting up your Web Server, you will have to copy these executables in the new Web Server's "cgi-bin" folder (you will have to make one if there is none).

IOGEAR
KVM • FIREWIRE • USB

USB 2.0 CardBus

GPU202

USB 2.0

40 times faster

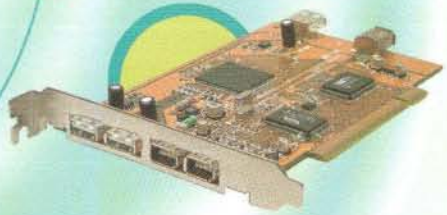
up to 480 Mbps

USB 2.0 PCI Card

GIC250U

USB 2.0 & FireWire PCI Combo Card

GUF220



4 Port USB 2.0 Hub

GUH204

For More Innovation Solutions Please Contact Us:

1-888-999-2836
sales@iogear.com
www.iogear.com

Also, it doesn't hurt to install your database software at this point if you don't have one already.

On Mac OS X, both Apache and OpenBase are installed by default.

To install WebObjects on any platform, you need to log on as the Administrator (or root) user. If you don't have Administrator (or root) access rights, it won't work at all.

On Mac OS X, you have to first **enable the root user** (which is **blocked by default**) and then to log on as the root user. To enable the root user, open folder /Applications/Utilities, and double-click on NetInfo Manager. Once NetInfo Manager is running, open menu Domain | Security | Enable Root User. You will be asked to enter a password for the root user. Now log off and log on again using 'root' as the user name, and the password you set previously. You are now logged as the root user and ready to install. After the installation is finished, you may disable the root user by running NetInfo Manager again.

Installation Steps

Insert your WebObjects CD in your CD-ROM drive. Double-click on the new icon that shows up in the desktop. Now double-click on the icon marked **WebObjects_X_Developer.mpkg**. This will call the installation program. Then, follow the instructions in the wizard-like window.

The following notes are meant to help you along the way. Read them before starting the install:

- The serial number is usually located on a sticker on the CD envelope.
- Be sure to read carefully the License Agreement for the next two hours. ;-)
- Most people should choose a Typical Install. The Typical Install includes everything except the following (generally unimportant) items:
- The Japanese language support
- The source code
- At one point, the WebObjects installation program could ask you for your Web Server's "cgi-bin" folder and its "docs" folder. Check them out and write them down before you start the install.
- This process takes a very long time - be patient.
- You will need to restart your computer at the end of the install, so it's a good idea to save your data and close all programs before even starting it.

Patches

I recommend that you install the latest patches for all of these (if you use them):

- WebObjects itself
- The operating system
- The Web server
- Your database software

For the Mac OS X, updating to the latest patch is very easy. You can get all the patches by going to the System Preferences panel, in the category Software Update. You will find there a button "Update Now". Clicking on it will check all software versions against an online database of patches. It will propose updates when they are available. I suggest you apply all of them. At the time this article was written, there was one patch available, which fixes many bugs. **Install it.**

On other platforms, updating to the latest patch may require that you visit each vendor's Web site and check for patches there. That's it — you are now ready to enjoy your development environment.

I didn't cover the deployment installation here because it is a complex topic that deserves its own article.

The Find-A-Luv example

I designed this column to have a good balance of theory and examples. I took good care of the theory, but I didn't want to make tiny useless examples and toy-code I had seen too often. Also, I didn't want to just show how to use the wizards of WebObjects and leave you alone with the customization. I needed a real Web Application and I wanted to develop it from scratch, not using just the wizards...

So throughout this column, I will be developing, as the main example, always the same Web Application. It features a Web Site for an imaginary dating services company. This company's promise is to help find nothing less than the client's soul mate. But be forewarned: if you're looking for your soul mate, this column will not help you.

We will start small and we'll expand this example as we go along.

Web Development Alternatives

WebObjects is not alone. There's a lot of competition out there. Many tools offer interactive Web Site/Server-side processing and Application Server solutions. Let's digest a few of the most popular ones.

Competing Interactive Web Site Solutions

The first ever solution for that is the CGI (common gateway interface). This is the most primitive solution. CGI is not a product; it's a feature of your Web Server that lets you insert your own program in you Web Site. That program can receive the user's input, and outputs a new Web Page based on it. The main disadvantage of the CGI is that the HTML code and database access are encapsulated into the source code; this is not pretty nor is it easy to develop. The main advantage is that you don't have to buy software to use CGI programs since it's already in your Web Server. Just use any computer language and you're on your way.

Sun Microsystems' Java Servlets are based on the same idea as CGIs, except they are written in Java and use their own module (called Servlet Runner program) that adds on to the Web Server. They have similar advantages and disadvantages as CGIs except for the fact that certain Web Servers don't support Servlets natively. Or if they do, their quality leaves a lot to be desired. This is the reason why commercial third party Servlet Runner programs (like Allaire's JRun) are available.

Allaire's ColdFusion is a commercial product that lets you write HTML "templates". That is, you write your HTML like you would normally do for your Web Page, and then you add in the same file the logic that makes the page interactive. You can even link your page to the database in only a few steps. The advantage is that you can use an HTML graphical editor for most of the work, and then add the logic afterward. ColdFusion is easy to use and produces results fast (just like HTML does). The main problem with ColdFusion is that it's not a real computer language; you can quickly arrive at the limit of what it can do.

Microsoft's ASP (Active Server Pages) also relies on "templates". But the logic is coded with Visual Basic. The main advantage is, like ColdFusion, fast development and tools that let you create pages easily. The disadvantage is that ASP is not portable: you can only use it on Microsoft's Web Servers.

Sun Microsystems' JSP (Java Server Pages) is an important player. JSP is based on the same idea as ASP, but the logic is coded in Java.

Here's a grid that summarizes the different characteristics of these solutions:

Alternatives / Characteristics	CGI	Servlets	ColdFusion	ASP/JSP	WebObjects
Comes with your Web Server	Yes	Some	No	Some	No
Uses templates	No	No	Yes	Yes	Yes
Logic and HTML in separate files	No, HTML in code	No, HTML in ccde	No, code in HTML	No, code in HTML	Yes
Flexibility of a real language	Yes	Yes	No	Yes	Yes
Supports most Web Servers	YES	Yes, with Servlet Runner program	Yes	NO	Yes
Uses Java	No	Yes	No	JSP only	Yes

Competing Application Servers

When Sybase wanted to issue an application server, they decided not to reinvent the wheel. They put together a set of existing tools then integrated them. Their Enterprise

Application Server (or EAServer) is composed of Sybase's Jaguar CTS (application server) and PowerDynamo (Web server). EAServer is integrated with development tools like PowerBuilder and PowerJ, but these come separately.

IBM's application server is called simply WebSphere Application Server (Standard Edition). It features Servlets and JSP, a Web Server, and integration with Visual Age for Java and WebSphere Studio — sold separately. The Advanced and Enterprise editions offer more features, like EJB, better performance, security and other advanced features.

The Oracle Application Server provides a basic set of features; Web Server (with Servlet and JSP support), EJB, a stable and scaleable environment for deploying Web applications. It is also integrated with Oracle's IDEs, Oracle Developer and JDeveloper.

BEA's WebLogic Server is an application servers that follows the J2EE standard to the letter. It features also Servlets and JSP for the dynamic HTML part, integrated with the most popular third-party IDEs. EJB, Java 2 Enterprise Edition and the latest standards are supported.

Here's a grid that summarizes the different characteristics of these solutions:

Alternatives / Characteristics	Sybase EAServer	IBM WebSphere	BEA WebLogicServer	Oracle Application Server	WebObjects
Uses J2EE/EJB	Yes	Yes	Yes	Yes	No
Works on Mac	No	No	No	No	Yes
Comes with an IDE & tools	No (separate)	No (separate)	No	No (separate)	Yes
Does distributed transactions, monitoring & load balancing.	Yes	Advanced Edition only	Yes	Yes	Yes

As I am writing this book, there are countless different application server packages on the market. I will not cover them all, but I am convinced that only WebObjects covers such a wide variety of tools and features. But in general, I have noticed that most of the other application servers are just a bunch of distinct tools put together. Many of these products are tied to a single Web server or database. On the other hand, WebObjects has fully integrated tools that work together as a whole and leverages whatever tools (Web Server, database) you already have. Its programming libraries and database tools are jewels and no other vendor offers similar tools. While WebObjects does not follow the J2EE specification, its proprietary nature enables Apple to provide easy-to-use RAD tools and many "value-added" features. Furthermore, WebObjects is the only application server available on the Mac OS X.

Sometimes, one just isn't enough...

DVIator for ADC

... and sometimes it is.

With DVIator and a DVI video card you can add additional ADC displays to a new G4, or upgrade an older Mac with Apple's new flat-panels.

Dr. Bott LLC

www.drbott.com • 877.611.2688 • 503.582.9944

By Marcelo Amarante Ferreira Gomes

Mac vs. Unix traditions

How to combine the best of both with MacOS X without gathering the worst

A Big Move

With the advent of Mac OS X, Apple has departed from the traditional Mac OS system software to a new model, that of Unix. This has brought a great impact, both upon users and developers. In this article, I'll concentrate on the impact it has on us, developers.

Mac OS X brings a lot of changes. We now have a single filesystem tree, for instance, instead of a tree for each mounted volume; we have lots of users, instead of the few we had with Classic, even though most are fake, like root or bin; we have a much more complex concept of process ownership, with process groups, effective and real user and group IDs and a controlling tty. These – and lots of other – changes also change the way we code, something that even resembles the move from 68k to PPC in 1994, but is a lot closer to the move from System 6 to System 7 in the late 80's and early 90's.

There were no new things to think of when we moved from 68k to PPC, only lots of programming pitfalls. But when System 7 came up, later to be renamed Mac OS 7, we actually had to think different. Those readers old enough to remember will recall that when System 7 came up, we had to think of High-level events, mostly Apple events, for simple tasks like opening a document, and try to use them whenever possible, instead of the low-level events and Finder lists we were used to. We had to revamp our user interface, so that our applications wouldn't look displaced in that new and way cooler-looking GUI. We had also to do away, once and for all, with our 24-bit stuff and use 32-bit-clean pointers and handles, paying attention to the possibility of the use of virtual memory in the system.

For those of you new to Macintosh, but experienced Unix developers, there were a few similar moves that the Unix community has gone through, too. They have been much smoother, since there was no single company dictating directions to developers, like Apple does in the Mac arena. To name a few, there was the move from the single mainframe with lots of terminals to lots of networked personal workstations; from UUCP and serial-based WAN links to TCP/IP and Ethernet-based LANs; from character-based UI to X-based GUI; from single-computer passwd files to distributed NIS and/or shadow user databases. Some of these were actually additional options, rather than changes. For instance, I hardly recommend the use of NIS, even though I do recommend shadow password files. And even today, I still recommend UUCP links and character-based UI in specific cases. But either the new features or the changes in them required some extra care in the way one writes a new piece of software, and I'm not referring only to programming pitfalls here.

I could go on, naming other changes like these, both in the Mac and Unix fields, but the important thing here is to point out that there are some major moves in development scenarios from time to time. And when such a move brings lots of changes at once, it also produces changes in the programming philosophy. That's what's happening with the introduction of Mac OS X. And those of us who fail to perceive this will probably also fail to survive in a market that is always hungry for state-of-the-art technology.

What's A Programming Philosophy, Anyway?

It's not only about programming caveats. The programming philosophy encompasses the way we design, we use, and even think about the uses for a computer system. Most Macintosh users think of their computers almost as a friend, while Unix users tend to think of them only as a tool to get some job done.

Mac OS has traditionally been very user friendly and highly customizable but very consistent through different applications, both in terms of the interface with the user and the functionality. Unix, while highly customizable in functionality – even more than the Mac in most aspects – is usually very harsh with its fixed user interface. It seems to have been designed to work as a server, while the Mac OS is a great workstation.

Veteran developers always take these facts into account, even if unconsciously, and write code that corresponds to their perception of what the user needs. What Apple is doing now with Mac OS X is bringing the two worlds together, releasing an

Marcelo Amarante Ferreira Gomes is an independent Macintosh, Unix and Internet consultant. He works most of the time with his Brazilian partners, and plays most of the time with his kids and/or his Lombard Powerbook that has the single most important OS in the world, besides most other junk in this industry. You can reach him at suporte@mac.com... if you're lucky. :-)

OS that is great for both a server machine and a workstation. This means that Apple wants users to have the same consistent user interface through all pieces of software they interact with, while having rock-solid system stability and compatibility.

Give Users What They Need

To build a successful Mac application, one must deliver what the user needs. Note that this does not mean that you should code just what the user expects. You can and should surprise the user with new and improved ways to do something. But try to invent features that users would guess when presented with a given screen. Developers with no previous Mac programming experience should observe mainly the Mac human interface guidelines. The typical Mac users expect every application to be similar to all others. They are used to reading the manual only as a last resort. Applications should be intuitive to them.

Apple has defined the recommended human interface guidelines required to achieve a successful Mac application long ago. These are constantly being improved, as new creative ways to interface with the users are developed, but the basic rules are always the same. What they say can be summarized as:

- when you add a feature to your software, do it in a way that resembles something you've seen before in another software;
- if the feature you're adding is radically new, try to imagine what the user would do to access that feature. Better yet, make a prototype of your application and have long-time Mac users play with it for some time. Pay attention to what they try to do, and listen to their comments very carefully.

There are lots of standard Mac OS interface elements that can be used to give users a clue to what they should do to get something done. If your particular feature does not fit into any interface elements, you should probably talk to a veteran Mac user about it. If you both decide there's no interface element that fits your needs, then try to invent your new element as similar as possible to another existing element. And always provide clear visual clues of the way your new element works.

Be Compatible – And International

In Mac OS, as well as in Unix, there are some conventions about where to put settings files, support code libraries, font files and lots of other things. Unlike in Unix, though, these special places should not be referred to by name. Apple has introduced the FindFolder system call a long while ago to give your software the path to these places. You can find the official Apple documentation on FindFolder at [1].

Under the Cocoa programming environment in Mac OS X, Apple thought you wouldn't need FindFolder, so they provided NSUserDefaults instead to store settings. See [2] for more info.

I see some of you asking: "why should I care about making a system call to find out what I already know?" This is a common pitfall. Settings files should go into the System Folder:Preferences folder, right? Not always. Under the Brazilian version of Mac OS,

for instance, they should go into the Pasta do Sistema:Preferências folder. Besides, always making sure to call the proper system functions enables Apple to restructure the filesystem layout without breaking any software and without having to support two folder structures at the same time.

With software that don't call FindFolder, international users end up with some settings files in the correct place while some other software create folders with foreign names and store their preferences there. Try to explain that to a user!

A much worse situation comes up with installer software. Say you need to put a certain file in a special folder, like System Folder:Extensions in Mac OS Classic, or to alter one of the rc.* under the /etc in Mac OS X. If you don't make the appropriate system call, your file may end up in the wrong place, and your software simply won't work. Please note that I, as a novice Mac OS X programmer that we all but a few Apple engineers are, haven't been able to find what the call would be to find the proper /etc directory or the rc.* files. Maybe it doesn't even exist, in which case Apple should provide us with such call.

Don't Mess Things Up

The same "don't mess up with the filesystem" can be said to wannabe Unix developers. Developers with no previous Unix experience must take into account various aspects of the Unix culture, especially those that have security implications. Apple itself has made some significant departures from traditional Unix practice.

For instance, most, if not all, Unix variants have a real directory for /etc. With Mac OS X, Apple has decided to make /etc be a symbolic link to /private/etc. There are other systems that put a symbolic link where a real directory would be, most notably Solaris with its /bin being a link to /usr/bin, but generally there's a good reason for this kind of change.

I don't know what was Apple's reason for putting a symbolic link in the place of /etc. The only thing I can think of is to make it easier to change most of the system's configuration by just making the link point elsewhere.

But this kind of change is dangerous, given standard Unix programming practice and system call side effects. It must be done at a very carefully chosen time during the boot process. It could also be done while the system is up, but one should be aware of the implications of this huge change in his or her particular system.

One problem that could appear stems from the fact that when a process opens a file, it gets a file descriptor that refers to whatever has been named by the time the open system call has been made. After that call, it doesn't matter if that file is moved, renamed or even deleted: the file descriptor sticks with the originally opened file. In case of deletion, the disk space is freed only when all of the processes that have opened it have also closed it. With all this, if one ever changes the symbolic link in /etc, there is the risk that one process, for instance, a server started at boot time, has a configuration file open that doesn't correspond to the configuration seen by all other processes started after the link has been changed.

The extent of damage caused by such situation can range from nothing at all to a completely disastrous crash. Or, worse yet, it can pose a security threat, for instance, when a client process thinks that the server will do some sort of security check, while the server thinks that check to be the client's responsibility.

Don't Reinvent the Wheel

Another departure Apple has taken from standard Unix with its Mac OS X was the use of the little known pax archiving utility for its installer process. I would personally have gone with tar, but I wouldn't mind Apple to have chosen cpio instead, since these two are the formats that most (99.9 percent?) die-hard Unix fans prefer to distribute software in.

With the use of pax, Apple has introduced some serious security flaws in their installer process. I have read a great article about this issue, but regrettably, I can't seem to find the URL in my bookmarks. Anyway, the main concern in that article is that pax does not deal very nicely with permissions and ownership data about already existing directories and files. Even worse, if you change a directory for a symbolic link, pax may get confused and delete the link, replacing it with a real directory. If the link had been placed there in order to save disk space and have the tree below it stored in another disk partition or other physical disk, you will run into big trouble.

Pax seems to have been designed to create all of the directories and files that it is installing, and it's not very clever to leave things like they are when it finds something already in place. That way, whenever you install new software, you risk introducing security holes that have long been corrected.

Until Apple fixes this, either by using tar or cpio instead of pax, or by making pax a little smarter, we will have to double-check our systems every time we install something. The easiest way to do so seems to be by looking at the installer package, taking note of all of the files it adds, along with the directories they reside in, and checking for their correct ownership, permissions and file type (regular file, directory, symbolic link, device file, etc.) Obviously, this work would be much easier if you take some time to store all of this information before the installation takes place and make any needed corrections afterwards.

Summing it All Up

Lots of other aspects about good development habits could have been mentioned here, both under Unix and Mac OS, but a magazine article is just too short to even give a good coverage, let alone being complete. The examples given here serve only to give you an idea of what kind of downturns you might run into by not being aware of standard practice.

But they also serve to give you an idea of what a programming philosophy is. It's about the way developers think, and the "right" way of doing things. Everyone will have a different idea of what is the "right" way of doing something, but some basic ideas are generally agreed upon among developers from a given OS. These basic ideas are what is being referred to here as a programming philosophy.

Quickly Build and Deploy Powerful Data-Driven Web Applications

Lasso Studio and Lasso Web Data Engine™ Lead The Way.

Building custom and shrink-wrapped database-driven Web applications requires a whole new way of doing things. Having pioneered the Web Data Engine™ over three years ago, Blue World and the Lasso Web Data Engine consistently lead the way providing a feature set Web developers describe as "incredible." Build online stores, discussion forums, resource management systems and other demanding database-driven Web applications with unrivaled performance, ease, security, extensibility, control and flexibility. Develop using multiple languages—including LDML, CDML, Server-Side JavaScript, Java and XML— and deploy across multiple platforms. Your Lasso code works identically regardless to which database you're connected. Lasso solutions for FileMaker® Pro databases easily scale to big iron ODBC-compliant databases like Oracle, Informix, Sybase and more with little or no change. What's more, the Lasso Java Application Programming Interface (LJAPI) provides developers an easy-to-use Java-based API for unprecedented extensibility.

Find out why hundreds of thousands of websites rely on award-winning Lasso technology for their business critical Web data. Download a 30-day evaluation copy at www.blueworld.com/download/ or order securely online today at the Blue World Store at store.blueworld.com.

Lasso Product Line – The leading Web tools for Macintosh and beyond.

bring business to the internet™

blueworld

How to Get There

Ok, we have to develop a new programming philosophy, that of Mac OS X, which should be a little different from Classic Mac OS and from Unix, but should not steer too far from either. But how do we get there? That depends on whether you are coming from Mac OS, from some sort of Unix or you are completely new to both. Either way, you've got to take a look at some web page from [3], especially [4].

Mac OS X for Classic Mac OS Developers

Of course Apple wouldn't succeed if they wanted us to rewrite all of our dependable code, that has cost us years of testing and debugging effort. But Apple can't stand still. So, they have provided us with three environments: Classic, Carbon and Cocoa. Ok, you've heard that before, so I'll skip to the important stuff.

Classic takes care of users. They won't have to buy new versions of all of their software title before moving on to X, but it's discontinued now, and it's not worth writing anything new for that environment. Cocoa, on the other hand, is the all-new innovative programming environment, in which all the good stuff is. Unfortunately for us who have invested on C, C++, Pascal or other languages, only Objective-C and Java APIs are available to work with it.

Then, there's Carbon. The environment for those coming from Classic Mac OS is definitely this one. Carbon is something in between Classic and Cocoa. It doesn't offer all the cool stuff that Cocoa offers, but it does offer most of it, including most of the coolest features we've been hearing about Mac OS X. It allows you to move your applications from the old Classic environment with little effort. Better yet, if you write for Carbon, your code may be able to run on Mac OS 9 and, depending on the set of system calls you choose, even on 8.x, by just having CarbonLib installed there. You can learn more about it in [5].

You can still think of files most the same way you did before, call most of your favorite system calls and think of the system mostly as if it really were the familiar Mac OS you've always known – for a while. It won't take long before you start cursing those most words in the previous sentence. You'll want to move on to Cocoa. Carbon is just a bridge to make that transition smoother.

And please try to stick to the rules. Learn the Aqua human interface guidelines (see [6]) and Quartz new API calls that are needed to implement some of Aqua look and feel (see [7] and [8].) Don't just use the same old ways of interfacing with the user. Some of the old methods, like custom MDEFs or WDEFs won't work, anyway.

Mac OS X for Unix Developers

By now, you already know that Cocoa is for you. Or isn't it? That depends on your previous knowledge of either Java or Objective-C or your willingness to learn a new programming language. If you already know any of these two languages, then look no further. You've got to go the Cocoa way, so read [9].

But if you are a seasoned C, C++, Pascal, Fortran, Basic, or something else programmer, you might want to give Carbon a try, at least for a while. Be warned, though, that you'll have a much harder time than your fellow Classic Mac OS programmers trying to go this way. While they already know most of the philosophy involved in writing good Mac OS software, you'll have to learn that from the beginning.

The Aqua API will be something new, and don't even think about writing an X-windows application. Most Mac OS X systems won't even have X-windows installed. But don't let that discourage you. The concepts involved are very similar. You still have a main() function calling an event loop and acting on events, just like on X-windows. Only the API will be new. Well, not only the API – you'll have to play by the Aqua rules. Read [6]; reread [6], then read it once again. Tired of reading? Read it backwards once more, just in case. It's never too much to stress how important the GUI rules are to keep Mac OS consistent.

"Damn, I could forget about all this GUI stuff and write a character-only app," I hear you saying. Ok, go ahead. But don't forget what your intended audience is. The average Mac OS user doesn't even know what a character-based UI is, let alone how to get to a command prompt. But if your audience is just a bunch of colleagues of yours and you do write a character-based application, please play by the traditional Unix rules, and try not to mess up with the System Folder or some other Mac OSisms brought over from Classic to Mac OS X.

Mac OS X for Newbies

If you've read the previous sections, you stand a good chance of already knowing which way to go. You don't know any programming language at all? Learn Java and/or Objective-C and go the Cocoa way. You already know some C and only want to know what this fruit-branded OS is about? Ok, try Carbon.

No matter what environment you choose to start working with, please, please, please! Follow the rules. All of them.

The Bottom Line

So, no matter what your background in some other OS is, if you don't have a good knowledge on both Unix and Mac OS standard practices, please reserve some time to learn about the culture on these systems. And don't try to save time on this! It will take lots of reading until you can say you're ready to write a successful application for Mac OS X. If at all possible, talking to veteran developers and users should take much of your time, too. But you will spend much more time, money and effort by going ahead coding, writing manuals, license agreements and other documentation, investing on marketing and distribution, only to find out that the users didn't like your software.

Worse than that, when many developers start coding without regard to standard practice, eventually some of them will come up with exceptionally good software that users will like and use although it doesn't follow standard practice. It may even happen with your title. But don't be happy with that.

When a significant number of software titles disregarding minimal guidelines start to gain acceptance, there will be nothing to call standard practice anymore. And Mac OS won't have the consistency it has today. Users will have a hard time figuring out how to use software; various titles will start being incompatible with one another. Then everything Apple and the Unix community across the globe have fought for and accomplished during many years will have been thrown away.

Apple fails to comply with traditional Unix practice in a few spots with dangerous consequences, but those trouble spots are simple to fix. They seem to be doing an overall good job and moving in the right direction. Let's hope that all – or at least most – of us will also get this right, and let's make Mac OS X a success by combining the best of Classic Mac OS with the best that Unix has to offer. And let's not pour in the worst by accident.

Acknowledgements

I'd like to thank Jacques do Prado Brandão, my friendly neighbor who, even not knowing a thing about computers or operating systems, was always there to help me revise some grammatical glitches that stupid word processors can't, as well as keeping a straight line of thought throughout the whole article.

Someone else to mention is Sergio de Souza Prallon, a fellow Brazilian consultant who has read – and understood – this article in its entirety. Being a Unix-only guy, he came up with the idea of writing the entire How to Get There section. Then he lied to me and told me it was good. :-)

And I could never forget about Neil Ticktin, MacTech's publisher. He has told me that the ideas I have presented here could make up a good article, thus giving the spark that ignited this whole thing. Has it blown anything up?

References

- <http://developer.apple.com/techpubs/mac osx/Carbon/Files/FolderManager/Folder_Manager/Functions/FindFolder.html>
- <http://developer.apple.com/techpubs/mac osx/Cocoa/TasksAndConcepts/ProgrammingTopics/UserDefaults/index.html>
- <http://developer.apple.com/techpubs/mac osx/index.html>
- <http://developer.apple.com/techpubs/mac osx/SystemOverview/devessentials.html>
- <http://developer.apple.com/techpubs/mac osx/Carbon/carbon.html>
- <http://developer.apple.com/techpubs/mac osx/SystemOverview/AquaHIGuidelines/index.html>
- <http://developer.apple.com/techpubs/mac osx/SystemOverview/QuartzPrimer.pdf>
- <http://developer.apple.com/techpubs/mac osx/SystemOverview/drawingwquartz2d.pdf>
- <http://developer.apple.com/techpubs/mac osx/Cocoa/CocoaTopics.html>

DEV DEPOT SM

PowerBook Essentials!

KEYSPAN Digital Remote

The Keyspan Digital Media Remote is a powerful infrared remote which allows you to control multimedia applications on your computer in the same convenient way that you now control your home TV. Great for PowerPoint, QuickTime, DVD players, CD players, and MP3 Players!

These are products you need, they're **kool gadgets** you won't find anywhere else, you'll get more out of your Powerbook if you have these!

PCMCIA Card Adds 2 USB Ports

Just slide this PC Card into your notebook and, instantly, you can begin connecting USB peripherals.

KEYSPAN USB Twin Serial Adaptor

Add two serial ports to your USB Macintosh! Simple to use, attractive on your desktop, and reliable!

Neoprene Case for iBook

A perfect fit! Easy to slip your iBook in and out of this 1/4" (6.5mm) Neoprene iBook Glove.

PO Box 5200 • Westlake Village, CA • 91359-5200 • Voice: 800/MACDEV-1 (800/622-3381)

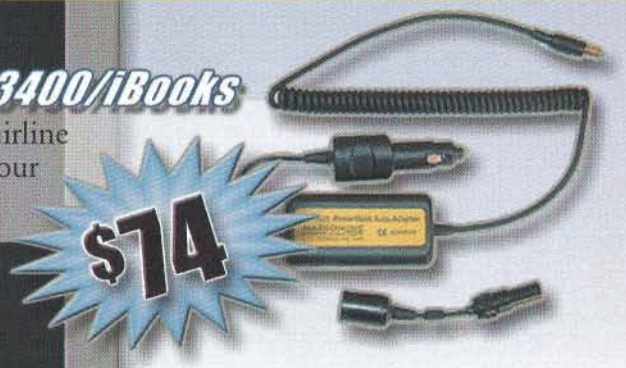
"World's Smallest AC Adapter" for almost all PowerBook and all iBooks!

AC Adapter for PowerBooks all G3s/iBook/3400

You spent how much to buy a portable computer? Then why do you haul a brick of a power adapter with you everywhere? This credit card sized power adapter makes your PowerBook truly mobile! Buy a second adapter for home!

You spent thousands of dollars to buy a portable computer, and now you haul a power adapter brick with you everywhere, but a second adapter for the office, buy a small credit card sized adapter for travel, this thing is kool!

Auto Air 50W Adapter for all G3s/3400/iBooks



Power your iBook or PowerBook in a car or on a commercial airline flight. Forget swapping batteries, watch DVD after DVD on your next long drive or flight!

iBook Bag

West Ridge Designs introduces the "Macintosh inspired computer bag," the iBookBag. Not just another pretty face, the iBookBag is built to protect!

Cool Pad Rotating Laptop Stand

Keep your laptop Cool! The CoolPad helps dissipate heat and letsyou rotate or elevate your laptop for a perfect fit!

Farallon SkyLINE Wireless 11 mb PCMCIA Card

AirPort compatible, the SkyLINE PCMCIA card turns your laptop into a wireless wonder! Send eMail, surf the web, or transferfiles through the AirPort base station or other AirPort enabledlaptops!

www.devdepot.com/powerbook/

Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

By Emmanuel Proulx

Deep Into the Request/Response Loop

Customizing Web Component Processing

PREFACE

This new column covers various advanced topics of programming Web applications with WebObjects 5. It is targeted towards knowledgeable WebObjects developers who are looking for that extra wisdom to help them go further.

In this first article, I introduce the Request/Reply loop, and how to customize it and control how Components are processed. I hope this will be valuable to you.

OVERVIEW

From the time a user clicks on a hyperlink to the time the page is displayed in the browser, lots of things happen. Of course, the browser and the Web server first initiate the communication and ask for a page. If you installed WebObjects with the CGI adaptor (as opposed to a native one), the following type of URL is used:

```
http://server.domain.subdomain/cgibin/WebObjects/ApplicationName
```

Here the WebObjects adaptor program is being called, and if there is an application called "ApplicationName" registered, it is being executed. WebObjects builds a page and returns it, calling your code to fill in the blanks. This part is very transparent. But the building of a page is not. How does it work? What happens? When does it happen?

Knowing the answer to these questions is important to expand, tweak and adapt your WebObjects program to the specific requirements of your target system. **Figure 1** illustrates how it works:

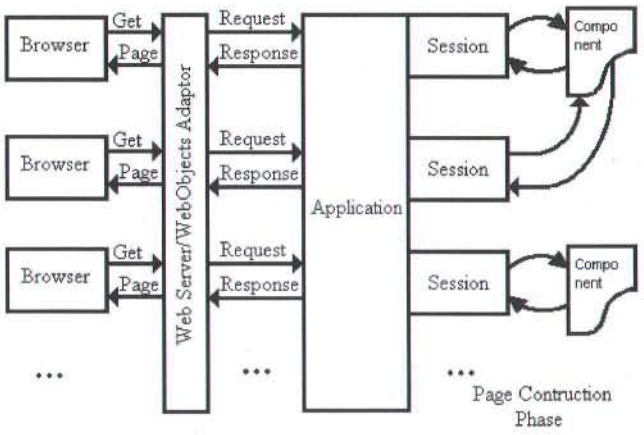


Figure 1. *Objects of the Request/Response Loop.*

The WebObjects adaptor program builds a Request object, which encapsulates the original "Get/Post" message received by the Web Server. This object is passed around the framework as WebObjects works on the Component (page template). Then WebObjects builds the Response object, which encapsulates the returned (pure HTML) page.

During this time, many classes of the framework call each other. You know about the Application and Session classes. You just learned about the Request, Response and Component classes. When they interact, they call different functions at different times.

Knowing when the functions are called and overriding the correct function is the key to customizing the request/response loop. **Figure 2** illustrates the different objects and methods and their interactions. The next sections describe them further, and prescribe when to overload them.

Emmanuel Proulx is a Course Writer, Author and Web Developer, working in the domain of Java Application Servers. He can be reached at emmanuelp@theglobe.com.

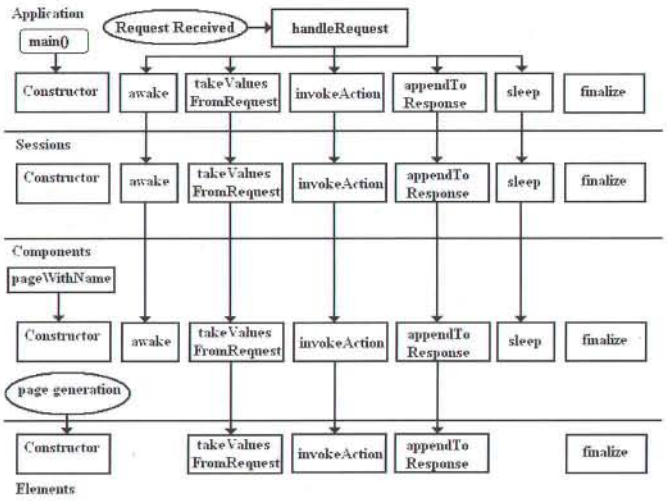


Figure 2. *Interactions Between Methods.*

Application Object

The Application class is a subclass of the virtual WOApplication class. Your Application class will only have a single instance per server. This class contains the main() function. By default, the generated code for this function calls WOApplication.main(), which creates a Singleton instance of your Application class. You have to use the following definition so your subclass will be recognized:

```
public class Application extends WOApplication
```

I have also stated that the Application object is accessible from all Sessions on the server, and its main use is to share data among them, and to hold global business logic. The Application's underling goal is to manage the Sessions and the request/response loop. Typically, you overload at the Application level if your code is general to all clients. That said, here are the main functions that you can overload to customize the framework to your needs:

- **main(String []) :** Overload to initialize variables before the system is started. Note that the default behavior is to call WOApplication.main(), which creates an instance of Application.
- **Application() (Constructor):** Ideal to put the system-wide initialization code.
- **handleRequest(WORequest):** Takes care of a single iteration of the request/response loop. The base implementation calls awake(), takeValuesFromRequest(), invokeAction(), appendToResponse() and finally sleep().
- **awake():**Called by the framework when there's a new request, before WebObjects begins processing it. Overload for example when you want to keep statistics of the frequency at which the system is being used.
- **takeValuesFromRequest(WORequest, WOContext):** The base implementation calls Session.takeValuesFromRequest(). As its

name stipulates, this function should be overridden when you want to process the request object from the application object.

- **WOResponse infokeAction(WORequest, WOContext):** The base implementation executes the action associated to a Request, by calling Session.invokeAction(). This function should be overridden if you intend to act upon the action itself (for example, to stop one from being executed).
- **appendToResponse(WOResponse, WOContext):** The base implementation calls Session.appendToResponse(). As its name says, this function should be overridden to add to the Response, or for any post-action operation.
- **sleep():** Called by the framework after a request/response is done with, before WebObjects waits for the next one. Overload for example when you want to keep statistics of the requests duration.
- **finalize():** The Application's destructor function is ideal to put the system-wide cleanup code.

Session Object

The Session object is a subclass of the virtual **WOSession** class. Your Session class has one instance per HTTP connection. WebObjects creates all instances for you, but you have to use the following definition:

```
public class Session extends WOSession
```

The main use of the Session class is to hold connection-specific information and business logic. Under the hood, it's also in charge of terminating itself and holding the Editing Context (database objects). You will notice that most of the functions you can overload are similar to the Application's. This is so you can decide the scope of your overloaded code. Typically, you overload at the Session level if the code is connection- or client-specific. The interesting functions are:

- **Session() (Constructor):** The constructor is called a single time shortly after WOApplication.awake() is called, but won't be called again for the current client. This function is ideal to put the connection-specific initialization code.
- **awake():** Called by the framework when there's a new request, during WOApplication.awake(), before WebObjects begins processing the request. The base implementation calls WOComponent.awake(). Overload to call request-specific initialization code.
- **takeValuesFromRequest(WORequest, WOContext):** Called during Application.takeValuesFromRequest(). The base implementation calls takeValuesFromRequest().in the page that was requested. This function should be overridden when you want to process the Request object from the Session object.
- **WOResponse infokeAction(WORequest, WOContext):** Called during Application.invokeAction(). The base implementation executes the action associated to a Request by calling invokeAction() in the page that was requested. This function should be overridden if you intend to act upon the action itself (for example, to stop one from being executed), depending on a client-specific state.
- **appendToResponse(WOResponse, WOContext):** Called during WOApplication.appendToResponse(). The base implementation calls appendToResponse() in the page that was requested. This function should be overridden to add to the Response, or for any post-action operation.
- **sleep():** Called by WOApplication.sleep() after a request/response is done with, before WebObjects waits for the next one.
- **finalize():** The Session's destructor function is ideal to put client-specific cleanup code.

Web Component

The Component objects are subclasses of the virtual WOComponent class. There can be multiple instances of a Component, and they are usually represented by local references.

WebObjects creates an instance of the Component with the name "Main" for you, returning it to the client browser, but you have to use the following definition:

```
public class Main extends WOComponent
```

You are responsible for creating instances of other Components. Usually, a Component is returned when an action is invoked. In that case, you create an instance of a Component by using this syntax:

```
WOComponent anAction() {
  return pageWithName("ComponentName");
}
```

The main use of the WOComponent subclass is to hold page-specific data and logic. On top of that, Components have the ability to generate the result page (using method **generateResponse()**). Again, most of the functions you can overload are similar to the Application's and the Session's. Typically, you overload at the Component level if the code is page-specific. The functions you can overload are:

- **Web Component Constructor:** The constructor is called when pageWithName() is called. This function is ideal to put the page-specific one-time initialization code. Use awake() to put page-specific per-request initialization code.
- **awake():** Called by the framework when there's a new request, during WOSession.awake(), before WebObjects begins processing the request. The base implementation does nothing. Overload to call page-specific per-request initialization code.
- **takeValuesFromRequest(WORequest, WOContext):** Called during WOSession.takeValuesFromRequest(). The base implementation calls WOElement.takeValuesFromRequest(), where WOElement is the root element (<BODY>) of the hierarchy of the page. The function is then called recursively on each element of the hierarchy. This function should be overridden when you want to process the Request object from the current page.

- **WOResponse infokeAction(WORequest, WOContext):** Called during WOSession.invokeAction(). The base implementation executes the action associated to a Request directly. This function should be overridden if you intend to act upon the action itself (for example, to stop one from being executed), depending on a page-specific state.
- **appendToResponse(WOResponse, WOContext):** Called during WOSession.appendToResponse(). The base implementation calls WOElement. appendToResponse(), where WOElement is the root element (<BODY>) of the hierarchy of the page. The function is then called recursively on each element of the hierarchy. This function should be overridden to add to the Response, or for any post-action operation.
- **sleep():** Called by WOSession.sleep() after a request/response is done with, before WebObjects waits for the next one. Ideal for per-request page-specific cleanup code.
- **finalize():** The Component's destructor function is ideal to put one-time page-specific cleanup code. Use sleep() to call per-request page-specific cleanup code.

Element Object

There's an object that I skipped on purpose here, the WOElement object. You usually don't subclass this class, unless you want to write your own customized elements. You can avoid this trouble by using custom components instead, which is easier to write. The interesting member functions are the constructor, takeValuesFromRequest(), invokeAction(), appendToResponse() and finalize(). The usage of these functions should be trivial.

Request Object

As stated before, the Request class encapsulates a Get or a Post HTTP request. The base class for WORequest is WOMessage, so some of the characteristics explained here are inherited from that base class.

The framework uses an instance of WORequest to pass it around during the pre-generation phase (awake(), takeValuesFromRequest() and invokeAction() functions). Here's a partial list of the information held in this class:

- A list of "cookies" (cookies are key/value pairs stored in the client's Browser). Keys are user-defined only. Cookies will be discussed in their own section.
- A list of all submitted form fields when applicable (Post request only). Form fields will be discussed later on.
- A list of "headers" (headers are key/value pairs containing the context of the request).
- The bits that make up the requested URL.
- Other request information (encoding, languages, protocol, request type).

I will skip cookies and form fields; they are covered later on. As for headers and the other information, there's no prescribed way of using these. It's up to you to figure out how to use them in your advantage. I can only show you how to access them. Following is an overview of the available functions.

Headers

Accessing headers is done with these functions of the class WORequest:

- **NSArray headerKeys ():** Returns a list of all available header names as Strings. Use these as keys in functions headerForKey() and headersForKey().
- **String headerForKey(String) and NSArray headersForKey(String):** These functions return the value (or values) of a header, when you pass the header's name (key). Use the first for single-value headers, and the second for multiple values.

But what are the fundamental request headers and what are they used for? Here's a link that lists some of the basic request headers:

http://www.w3.org/Protocols/HTTP/HTRQ_Headers.html

But there may be other headers, because some are specific to the Web browser. To find out about those extra headers, the first thing that comes to mind is to overload takeValuesFromRequest() in one of Application, Session or Main, and print them out. The following piece of code does exactly that:

```
public void takeValuesFromRequest(WORequest r, WOContext c)
{
  super.takeValuesFromRequest(r,c);

  if(r.headerKeys() == null) return;
  for(int i=0; i< r.headerKeys().count(); i++) {
    String key = (String)r.headerKeys().objectAtIndex(i);
    NSArray values = r.headersForKey(key);
    System.out.println("Found header " + key +
      ": " + value.toString());
  }
}
```

Here's what the output might look like:

```
Found header: accept-charset = ("iso-8859-1,*,utf-8")
Found header: accept-language = (bg)
Found header: accept-encoding = (gzip)
Found header: connection = ("keep-alive")
Found header: user-agent = ("Mozilla/4.6 [en] (WinNT; I)")
Found header: host = ("localhost:3878")
Found header: accept = ("image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, image/png, */*")
Found header: referer =
("http://www.imaginarypenda.com/gotosite.html")
```

Dismantling URLs

Next, let's have a look at the functions returning the parts of the requested URL. Let's use an example for each function. Imagine the user wrote the address:
http://www.imaginarypenda.com/cgi-bin/WebObjects/App

Here are the methods of WORequest to take a URL apart:

- **String uri():** Returns the last part of the whole URL, from the first slash to the end. For example, if the user typed the URL above, this method should return "/cgi-bin/WebObjects/App".
- **String adaptorPrefix():** Provides the adaptor's name. For example it could be "/cgi-bin/WebObjects/" on Mac OS X or Unix, or "/cgi-bin/WebObjects.exe/" on Windows. This example shows the CGI adaptor prefix; it may be different if using a native Web server adaptor.
- **String applicatioNumber():** Returns the user-requested application number, when provided. This information is usually not provided by the user in the URL (as not shown above). In these cases, this functions returns -1 and meaning any one instance of the application was called.
- **String applicationName():** Returns the application's name. In our example, it is "App".

Let's dissect some URLs and see which function return which part. Here are three examples.

- The first example below shows a MacOS X-based server, pointing to any instance of Find-A-Luv (no instance number). In this case, applicationNumber() will return -1.
- The second is a Windows-based server, pointing to any instance (we could have skipped the -1).
- The third is a Unix-based server, with native adaptor (no CGI involved), and pointing to the third instance.

```
http://www.aUrl.com  /cgi-bin/WebObjects/
Find-A-Luv
http://www.aUrl.com  /cgi-bin/WebObjects.exe/
-1 /  Find-A-Luv
http://www.aUrl.com  /WebObjects/    3    Find-A-Luv
  adaptorPrefix()   applicationNumber()    applicationName()

  uri()
```

Other Request Info

Following is an brief overview the "other information" functions:

- **method()**
  StringAny valid HTTP method, like "GET", "PUT, "POST", "HEAD", etc. See http://www.w3.org/Protocols/HTTP/ Methods.html for a complete list of methods and their usage.
- **browserLanguages()**
  NSArray of String The list of languages (see your browser's language preferences).

- **content()**
  NSData Always null unless there was raw data posted with the request.

- **httpVersion()**
  String For example, "HTTP/1.0".

- **userInfo()**
  NSDictionary A set of key/value pairs, passed around during the processing of the request. Empty by default, but you can use it to convey data around the framework.

I don't want to spend too much space on these functions since their usefulness is limited.

Response Object

The Request's counterpart is the Response object, which encapsulates the page returned to the requesting browser. The WOResponse class is also a subclass of WOMessage, so some of the characteristics we'll cover come from that base class.

Typically, the Response object is created by the framework and passed around the elements hierarchy during the generation phase. Then it is passed to the Component, Session and Application objects, during the post-generation phase. All of this is accomplished by function appendToResponse().

The information in this object is very similar to the Request object. The difference is that you can change this information, and write to the buffer holding the returned page. Here's an overview of the kind of information you can get and set:

- A list of cookies. Cookies will be discussed in their own section.
- A list of resulting headers.
- The actual contents of the returned page.
- Other request information (status, encoding, protocol).

Response Headers

We saw how to get headers in previous sections. The same getter functions exist in the WOResponse object (headerKeys(), headerForKey(), headersForKey()). On top of that, you can set the headers returned to the browser:

setHeader(String value, String key) and setHeaders(NSArray values, String key): These functions fix the value (or values) of a header, given its name (key). Use the first for single-value headers; use the second for multiple values.

A list of basic response headers can be seen at: http://www.w3.org/Protocols/HTTP/Object_Headers.html

Response Text

For getting and setting the contents of the returned page, WOResponse has many methods. Here are a few of them:

- **NSData contents():** The raw bytes that constitute the returned HTML. An NSData object encapsulates an array of bytes (byte []), and the interesting functions are bytes() and length().
- **setContent(NSData):** Replaces the whole returned page with the one you provide.
- **appendContentString(String):** Adds the specified string to the end of the resulting page without changing anything.
- **appendContentHTMLString(String):** Adds the specified string to the end of the page, but escapes the HTML-specific characters. For example, the character '<' will be changed to '&lt'.

WARNING: you usually don't want to use setContent() because it wipes out the previously computed HTML. Use the append...() methods instead.

Other Response Info

The "other information" methods of WOResponse include:

- **defaultEncoding()**
  setDefaultEncoding(int) Gets and sets the default encoding, specifying the character set that should be used by the returned contents.

- **contentEncoding()**
  setContentEncoding(int) Gets and sets the encoding, specifying the character set that is being used by the returned contents.

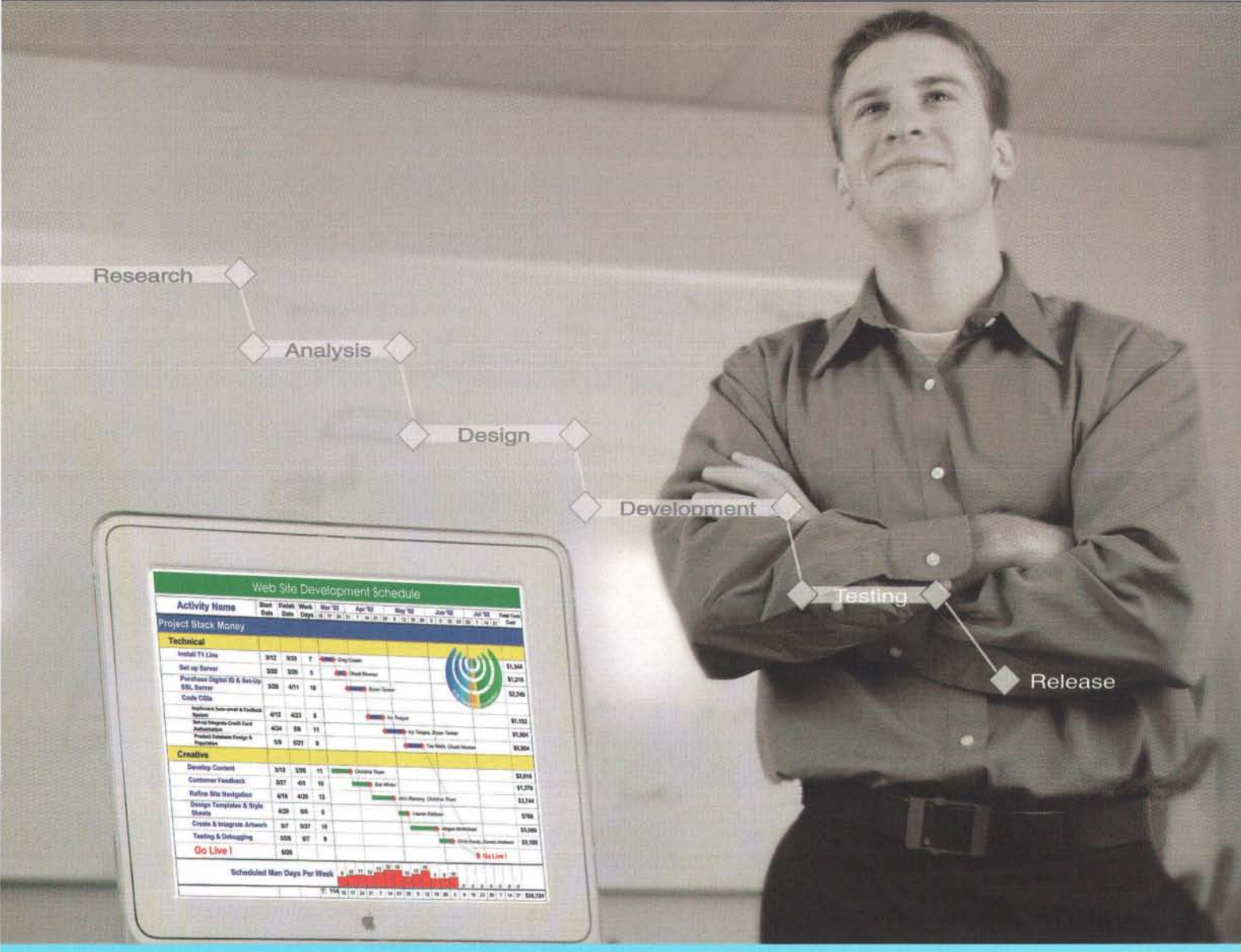
- **httpVersion()** Gets and sets the a string indicating the protocol format

- **status()**
  setStatus(int) Gets and sets the status, indicating if the generation was successful (status() returns 200) or if an error occurred. See this page for a list of status codes: http://www.w3.org/Protocols/HTTP/HTRESP.html

- **userInfo()**
  setUserInfo(NSDictionary) A set of key/value pairs, passed around during the processing of the response. Empty by default, but you can use it to convey data around the framework.

Conclusion

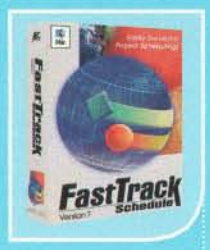
As you've seen here, there is more to a request/response loop than simply asking for a page and getting the HTML back. Many objects and methods are called along the way. As we've seen, most of the methods can be overloaded to intervene in the loop at a specific moment. We've also looked at the ways to manipulate the request and response themselves. Again, what to do with this knowledge is up to you, but I can almost hear your brain pondering. MT



One incredibly successful project manager.

What does this guy know about software development that you don't? He knows that with FastTrack Schedule 7.0 he'll shave hours off the time it takes to organize, track and manage the details of all his development projects. And his eye-popping schedules are sure to turn heads and get results.

Whether the project on tap is coding the next killer app or putting a fresh coat of paint on a client's website, FastTrack Schedule's three powerful views display your information the way you want—as a schedule, a calendar, or as a resource graph that tracks the people, equipment, and materials crucial to project success.



And with our new compatible Palm OS version, you can sync schedules between your desktop and handheld. So even when you're on the run, your schedules are right at your fingertips. FastTrack Schedule also includes FastSteps™ intra-application scripting, full support for AppleScript, and is available in a compatible Windows version. For a free demo version or to order, call us today at **800.450.1983** or visit **www.aecsoft.com**.

Easily** the best in **Project Scheduling!

AEC SOFTWARE FastTrack Schedule Version 7 FastTrack Schedule for Palm OS Details

finalist macworld eddys

MVP Winner PC Computing

5-STAR PC Computing

By Andrew C. Stone

Internationalizing Cocoa Applications - A Primer for Developers and End Localizers

Now that you have written that amazing object-oriented Cocoa application, it's time to bring the fruits of your creativity to the rest of the planet. This article has two components: the steps a developer needs to take to allow the application to be translated, and the steps the translator performs to localize an application to another language. What sets Cocoa localization apart from other development environment localization systems is its pure simplicity. Once an application has been prepared to be localizable by a developer, any number of languages can be added to the final product, without any recompiling by the developer or any access to the source by the parties who are doing the translations. In fact, the localizer can be a regular end user with a love of native language Mac OS X applications! These facts signify that you can ship your application in your native language, and then begin the process of internationalization afterwards. We'll examine what the developer has to do, and then the steps required by the localizer.

Making An Application Localizable

Translatable text in a graphical user interface appears in two places: the Interface Builder files (nibs) which display menus and interfaces, and the embedded strings that are used in programmatically-created alert and dialog windows. A developer who follows a few guidelines will actually have no additional work to perform to localize their application:

- Make all programmatic text, alert panel's titles, messages and buttons use translation macros instead of quoted English strings
- Always add interface files as localizable resources

Translatable Dictionaries - the .strings files

As you write your code, whenever you use a string that's going to be displayed to the user, use one of the NSLocalizedString macros (NSLocalizedString, NSLocalizedStringFromTable, NSLocalizedStringFromTableInBundle). When your code is executed, these macros look up the string in a dictionary file with a ".strings" file ending. The .strings file is generated by a command line program, genstrings, that processes your code looking for the macros. Thus, there are three steps to creating the .strings files that get added to the English.lproj (we'll use English as the native language in this article, although you can develop in any language using Project Builder):

- Use the NSLocalizedString* macros in your code
- Run genstrings to generate the .strings files
- Add the generated files to your PB project as localizable resources

Here is a sample line of code, a message to set what the Undo menu displays once this action has been performed, which will appear in the user's chosen language:

```
    [[self undoManager]
setActionName:NSLocalizedStringFromTable(@"Change Print Info",
@"Muktinath", @"Action name for changing print info.")];
```

NSLocalizedStringFromTable() takes the string to be displayed ("Change Print Info"), the table to look for the string in (Muktinath.strings), and a helpful comment for the person who is translating the phrase.

Running genstrings on this code would produce an entry in the Muktinath.strings table:\

```
/* Action name for changing print info. */
"Change Print Info" = "Change Print Info";
The translator would then make a copy of the .strings file and
translate the right hand side. For example, en Espaing i
/* Action name for changing print info. */
"Change Print Info" = "Cambiar Info de Imprimir";
```

When using the macros in the standard alert panel, one technique to make your code easier to read is to use #defines, which also allows easy reuse:

```
#define TERMINATE_TITLE NSLocalizedStringFromTable(@"Quit",
@"PackIt", "Title of alert panel")

#define TERMINATE_MSG NSLocalizedStringFromTable(@"Remove
temporary files?", @"PackIt", "Message when temp files exist")
```

Andrew Stone <andrew@stone.com> is founder of Stone Design Corp <http://www.stone.com/> and divides his time between farming on the earth and in cyperspace.

```
#define CANCEL NSLocalizedStringFromTable(@"Leave alone",
@"PackIt", "Button title to not remove the files")

#define OK NSLocalizedStringFromTable(@"OK",@"PackIt", "Button
choice for OK'ing a requested operation.")
...
  if
(NSRunAlertPanel(TERMINATE_TITLE,TERMINATE_MSG,OK,CANCEL,NULL) ==
NSAlertDefaultReturn)
  ...
```

Generating the Strings files: genstrings

Once you have removed all direct uses of English strings in your program, then you are ready to collect all the translatable strings into their .strings files.

```
Type "genstrings" in a terminal window to see its options:

Usage: genstrings [-j] [-s <routine-name>] [-o <output
directory>] file1.[mc] ... filen.[mc]

Help: genstrings generates .strings file(s) from your source
code.
      The names of your source files are the arguments: file1
... filen.
      * C and Objective-C:
        Source lines with NSLocalizedString(string, comment)
will
          generate an appropriate string table entry on
Localizable.strings.
        Source lines with NSLocalizedStringFromTable(string,
tablename, comment)
          will generate an appropriate string table entry in
tablename.strings.
        Source lines with
NSLocalizedStringFromTableInBundle(string, tablename, bundle,
comment)
          will generate an appropriate string table entry in
tablename.strings.
      * Java:
        The -j option sets the expected input language to Java.
In this case the above
          keywords are changed to Bundle.localizedString,
Bundle.localizedStringFromTable,
          and Bundle.localizedStringFromTableInBundle (instead
of the Objective-C defaults).
      The -s option substitutes its argument for
NSLocalizedString.
        For example, -s MyLocalString will catch calls to
MyLocalString
        and MyLocalStringFromTable.
      The -o option specifies what directory tables should be
created in.
```

A simple invocation for a multi-level Objective C project would be something like:

```
genstrings *[hmc] */*[hmc] */*/*[hmc]
```

All source files in the top three levels would be searched to produce the output strings file. Add the produced .strings files to your resources folder in Groups & Files in Project Builder, and then make them localizable, as described in the next section.

Making Files Localizable

The developer's second major job is to add all translatable Interface Builder files as "localizable". In Project Builder:

- 1. Add the new interface file (Project -> Add Files... or just drop it in to Groups & Files outline view - I like to place .nibs into "Resources".

If you have saved the .nib file in English.lproj - then steps 2. and 3. are not necessary, otherwise:

- 2. Project -> Info to bring up interface inspector
- 3. Click "Localization & Platforms" popup - choose "Make Localized"

Adding other language versions of the interfaces is done with Project->Info with the file to localize selected.

This moves the interface file to the English.lproj in our example. To add a new language version, select the interface file in Groups & Files, choose "Add Localized Variant..." from the "Localization and Platforms" popup and type in the name of the language in English (example, for Espao the English.lproj in our example. To add a new language version, select the interface file in Groups & Files, choose "Add Localized Variant..." from the "Localization and Platforms" popup and type in the name of the languager of the user's language preferences, if those languages exist in the project. The same procedure applies to .strings files and other resources that require localiization. The localizer now has the resources needed to translate the program.

End User Localization - Anyone can do this!

Localizing an application can be done by anyone who has a basic knowledge of Interface Builder, TextEdit, Terminal and how to open file packages in Finder. Interface Builder comes on the Developer CD that shipped with the original OS X 10.0, so you'll need to install the Developer package if you haven't already. A translator doesn't even need to know how to use InterfaceBuilder if the developer extracts the localizable strings from the interfaces with a special tool, **nibtool**, described in the Advanced section below.

First, copy the application you wish to localize, just in case you mess something up! Since you will be editing files inside of the Application wrapper, you may to set the permissions so that you'll be able to save the changes:

1. Launch /Applications/Utilities/Terminal
2. Change directory to where you copied the application
cd <DIRECTORY>
3. Change permissions to allow saving:
chmod -R a+rwX <APPLICATION>.app

Now, you need to open the application wrapper. In Finder, hold down the control key while clicking on the application file and choose "Show Package Contents" from the contextual menu:

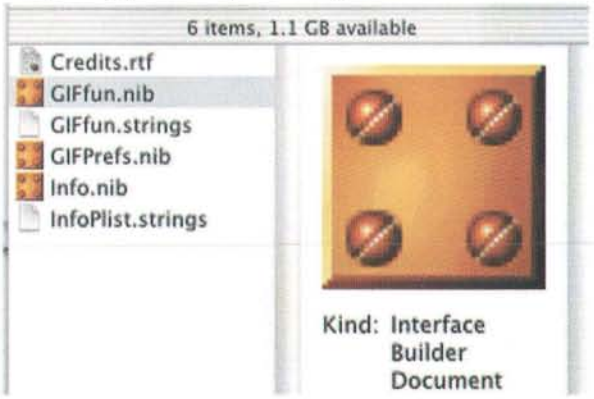
Control-Click an application to open it up

To test your application, set the system language to the one you are localizing. In System Preferences, International Pane, drag the language of choice to the top of the list:

Be sure to set your language before you launch our test application!!

An OS X application has one folder, "Contents". Inside of that folder is "Resources". In Resources, you'll find English.lproj. This folder should be duplicated, and then renamed to the English version of your language, e.g. Spanish.lproj, Dutch.lproj, French.lproj, Japanese.lproj, German.lproj, Italian.lproj, etc.

Now you can edit all the files in your language's .lproj directory and test your application by launching it anew!



There are three types of files to edit: .strings, .nib, and .html/.rtf informational/help files.

The .strings Files

The most boring task is to convert the strings which appear programmatically: the titles, messages, buttons from alert panels, and the "action names" that the Undo system uses to keep a track of what it's undoing. (For example, if you change the way the page was layed out, the Edit menu would display "Undo Change Print Info".) Sophisticated software like Stone Design's Create® may have hundreds of operations that can be performed, and require translation. Smaller applications may have very few of these, so don't get discouraged!

These strings appear in .strings files in the .lproj directories. You can use the free, Apple-provided, and Cocoa-based TextEdit program to edit these files; TextEdit supports unicode, the lingua franca of the Cocoa Text system, so all of the special characters found in other languages will be correctly preserved. Because the .strings extension is not registered with TextEdit, double-clicking a .strings file brings up the 'There is no application available to open the document "Something.strings"- "Choose Application"' dialog. To avoid this, place TextEdit in your dock, and drag the .strings file on top of the TextEdit icon to quickly open the file.

On the left hand side of a typical entry in the .strings file, the development language string appears, in quotes, followed by an equals sign. The right hand side of the entry is to be replaced by the equivalent phrase in your language. Finally, because these files need to be easily parsed by the application, the entry ends in a semi-colon, and uses /* and */ to delimit comments:

```
/* Name of Resource Source - like Patterns */
"Pattern" = "Pattern";

So, the French localizer would translate the second half into:

"Pattern" = "Texture";

And the Spanish localizer:

"Pattern" = "Motivo";

And the German localizer:

"Pattern" = "Muster";
```

One note about translating strings is the occurrence of format strings within the quotes. The programmer can use %@, %d, %f and other "sprintf"-style placeholders to place runtime information in the programmatically-generated text. For example, the following string will display the name of the pattern:

```
/* alert message to prevent removal of default pattern */
"You cannot delete the %@ pattern." = "You cannot delete the %@
pattern.";
```

When the program is running, the message would replace the %@ with the actual name of the pattern that cannot be deleted.

The localizer must take care to preserve the number and order of these placeholders. Programmers should make this easier for translators by providing info on the parameters in the comments. In the case of multiple parameters, the parameters are passed in order, and localizers need to keep this mind, even if it means awkward sentence structure, because otherwise, the program will crash! Be careful.

To insert quotes into the quoted string prefix the quote with a backslash: "The word \"quotes\" is quoted." To insert a single, literal "%" in a string, use "%%".

Besides the genstrings-produced .strings files, there is also an InfoPlist.strings file which has a few strings that can be localized:

```
{
    CFBundleGetInfoString = "Stone Design's Create®. © 1990-
2001, Stone Design Corp. Visit www.stone.com";
    NSHumanReadableCopyright = "© 1990-2001, Stone Design
Corp.\nVisit www.stone.com";
  // Document type human-readable names.
    "Stone Design Graphic Format" = "Stone Create (cre8)";
    "NSPDFPboardType" = "Portable Document Format (PDF)";
    "NSTIFFPboardType" = "Tagged Image File Format (TIFF)";
    CFBundleHelpBookName = "Create Online Manual";
    CFAppleHelpAnchor = "CreateHelp";
}
```

By translating the file types such as "NSPDFPboardType", you can have more attractive popups in your Save panels. The CFBundleHelpBookName key controls the display title of the Apple Viewer help your application provides and should be translated.

The Interfaces - Using Interface Builder

Now comes the fun part! Interface Builder lets you open and edit the .nib (NeXT Interface Builder) files, which are the actual interfaces used by a program. To give yourself a real boost, first translate the nib that contains the main menu, usually named the same as the application, e.g. "GIFfun.nib":

The process of converting the IB files goes like this:

for each IB file do:

- For each string you see, double-click it to edit, and replace with a translated word or phrase
- If necessary resize the control according to the feedback provided by IB
- If an interface element has a help tip, translate that as well Save and test by quitting and relaunching the application

A few fine points:

- You can tab between menu and matrix items to increase editing speed
- To set the title of a window, click on the window, select Tools->Show Info, and select "Attributes" in the popup
- Note that IB (and OS X) requires that you hit <RETURN> or <TAB> to make your edits "stick".
- Don't forget to translate each pane in a TabView: Double-click each tab to make its pane visible

Each interface element can have an attached Help Tip - those cute little yellow windows that pop up if you leave your mouse over a control for a few seconds that describe the functionality of the control. To change these:

- Select the control
- Tools->Show Info, "Help" popup pane
- Enter the tip in your language, and be sure to hit <RETURN> to make it stick

Advanced Localization Topics: Bundles, nibtool and Apple Viewer Help

As usual, the simple life ain't so simple! One thing to watch for is good engineering on the part of the developer.

Cocoa is a dynamic, runtime bound system, which means that objects (code and interfaces) can be loaded upon demand, not when the program starts. In Create®, for example, there are 50 interfaces which are dynamically loaded. This means very fast launch times, and much better memory usage, because you only load resources that you use as you use them. These "bundles" are layed out in a similar manner to the packaging of the application; each *.bundle* folder contains a Contents folder, which has the Resources folder, which contains an English.lproj with all localizable resources for that bundle. Most bundles just have one nib file — but you need to repeat the process described above (duplicate the English.lproj, rename it to your language in English, and localize the files in the <LANGUAGE>.lproj) for each .bundle folder in the application's Resources folder.

With the introduction of InterfaceBuilder 2.1, Apple has provided localization capabilities to the command line utility named **nibtool** (/usr/bin/nibtool). **nibtool** can extract the localizable strings in a nib to a .strings file, which can be translated, and then reincorporated into a copy of the nib file. This will not adjust spacing of the UI elements so it's just a start, but it might help things go faster.

To generate the strings file for a particular nib file:

```
nibtool -L myfile.nib > file.strings
```

file.strings will now contain entries such as "key" = "key" and be in Unicode (UTF-16) format. Next, give the resulting .strings file to a translator and have them convert the second "key" entry to "key in other language"

To reincorporate these translated strings:

```
nibtool -d file.strings -w newLocalization.nib myfile.nib
```

nibtool will take the contents of file.strings and replace "key" in myfile.nib with "something in other language" in the translated version "newLocalization.nib".

The final, and most demanding, part of the translation is the Online Help system - the folder of linked .html files that contains a special "Sherlock-style" index for instant searching. Stone Design keeps our help files in Create®, the localizer then edits the help directly, and produces the HTML with a simple export. This is usually easier than hand-hacking html because you also want to change the screenshots to reflect the translated interfaces. Please read my MacTech article of a few months ago entitled "Help On The Way! A guide for the perplexed on adding Apple Help to your OS X application" for complete instructions on adding online help to an application.

Conclusion

OS X has complete support for internationalization and is simple enough that end users can add new languages. With a small amount of effort, developers can produce applications which can be translated to new languages out in the field, without access to source code, by regular yet adventurous users.

By Tim Monroe

F/X

Using Video Effects in QuickTime Movies

Introduction

The *QuickTime video effects architecture*, introduced in QuickTime 3, is an extensible system for applying video effects to single images or video tracks (called *filters*), and to pairs of images or video tracks (called *transitions*). QuickTime includes an implementation of the 133 standard transitions defined by the Society of Motion Picture and Television Engineers (SMPTE), as well as some additional effects developed by the QuickTime team. The SMPTE effects include various forms of wipe effects, iris effects, radial effects, and matrix effects. Of all of these, my personal favorite is a wipe effect called the *horizontal barn zig-zag*, shown in **Figure 1**.

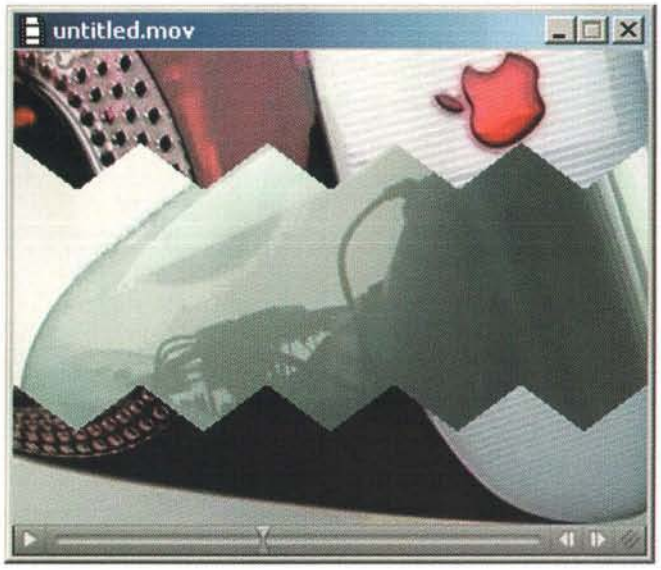


Figure 1: *The horizontal barn zig-zag wipe effect applied to two video tracks*

The additional QuickTime effects include transitions like a simple *explode* (where the first image is exploded outward to reveal the second image) and a *push* (where the first image is pushed aside by the second image). **Figures 2** and **3** show these effects applied to two penguin images.

Figure 2: *The explode effect applied to two images*

Tim Monroe's lizards have perfected the color-tint effect, changing from green to brown (and vice versa) at their whim. In his day job, Tim is a member of the QuickTime engineering team. You can contact him at monroe@apple.com.

Top 10 Reasons Why You Should Buy Funnel Web®

10. It's the most accurate Web site visitor analysis tool available
9. Everyone should own at least one product named after the world's deadliest spider
8. One of the developers once worked in the gear room of the world's highest aerial tram
7. Buy the product and we'll let you keep the CD
6. It produces reports so beautiful you'll want to carry them around in your wallet
5. It runs on Mac, Linux, Windows, FreeBSD and Solaris
4. Someone needed to start a new trend
3. Thousands of your colleagues are already using it
2. It's 400% faster than competing products*

And the #1 reason why you should try Funnel Web Analyzer is...

(What, you think we're going to give it up on the first date?
Go to www.quest.com/top10 to find out!)

*Refers to a recent independent review from Neiger Computer Consulting.

QUEST SOFTWARE
www.quest.com

Funnel Web Analyzer is one part of our complete suite to map, manage and measure your Web site.

United States 1.949.754.8000 • Germany +49.221.5694.1111 • United Kingdom +44.1628.601000 • France +33.1.4131.96.96 • Australia +61.3.9811.8000

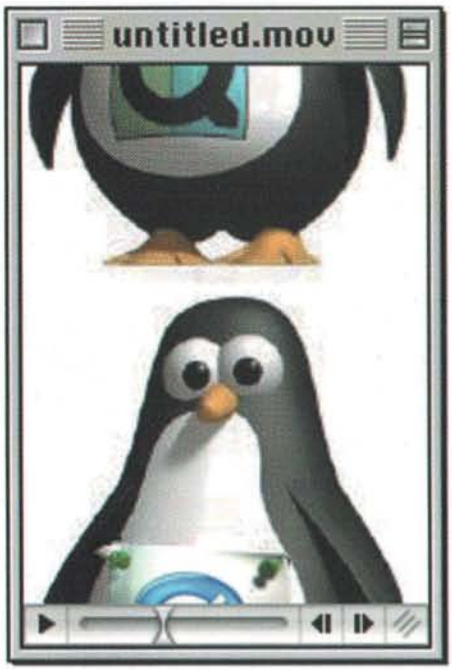


Figure 3: *The push effect applied to two images*

QuickTime also includes a very nice *cross-fade* or *dissolve* transition (which produces a smooth alpha blending from the first image to the second) and a nifty *film noise* filter that makes a video track look like old, faded, dusty, and scratched film. **Figure 4** shows a frame of a movie with the film noise effect.

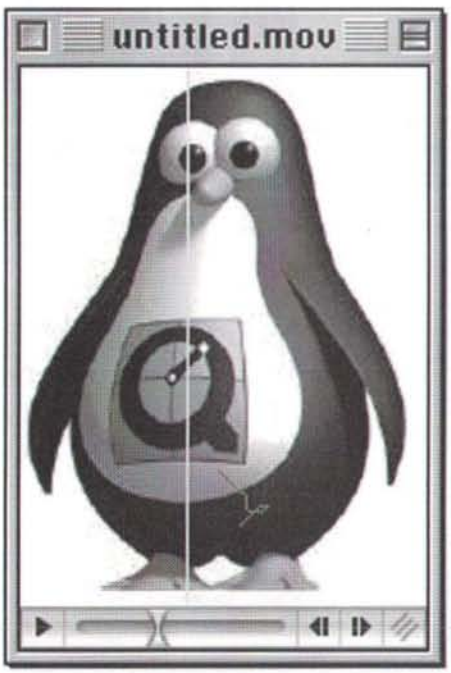


Figure 4: *The film noise effect applied to a movie frame*

There are several video effects that operate on no source images or video tracks at all, called effects *generators*. For instance, we can use the *fire* effect to generate a real-looking fire (see **Figure 5**), and we can use the *cloud* effect to generate a wind-pushed, moving cloud. With generators, we will usually want to composite the effect onto some other image or video track. **Figure 6** shows the fire effect composited onto the penguin image. (Ouch, that's gotta hurt!)

Figure 5: *The fire effect in a movie*

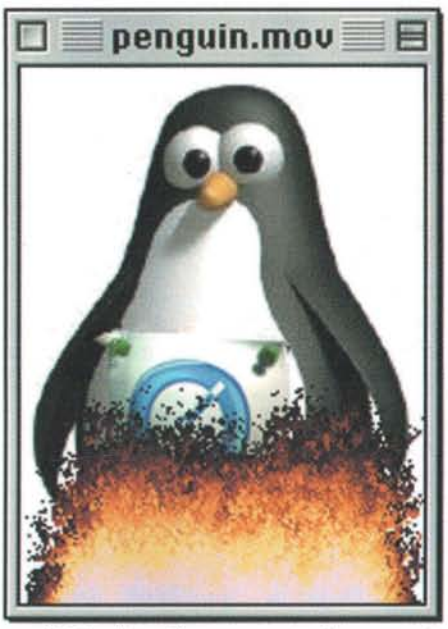


Figure 6: *The fire effect composited onto an image*

The data describing an effect is stored in a video track, and the actual effect itself is generated in real time as the movie is played. These effects use extremely little data to achieve the desired visual output. For instance, a video track that specifies the fire effect is only about 60 bytes in size; when the track is played, QuickTime generates a real-time, non-repeating, dynamic fire image.

Generators, filters, and transitions are implemented in the general QuickTime architecture as *image decompressor components* (of type decompressorComponentType). One thing this means is that we can reference a specific effect by providing a four-character code, which is an image decompressor component subtype. Here are a few of the available effects types:

```
enum {
   kWaterRippleCodecType          = FOUR_CHAR_CODE('ripl'),
   kFireCodecType                 = FOUR_CHAR_CODE('fire'),
   kFilmNoiseImageFilterType    = FOUR_CHAR_CODE('fmns'),
   kWipeTransitionType            = FOUR_CHAR_CODE('smpt'),
   kIrisTransitionType            = FOUR_CHAR_CODE('smp2'),
   kRadialTransitionType          = FOUR_CHAR_CODE('smp3'),
   kMatrixTransitionType          = FOUR_CHAR_CODE('smp4'),
   kCrossFadeTransitionType     = FOUR_CHAR_CODE('dslv'),
   kPushTransitionType            = FOUR_CHAR_CODE('push')
};
```

Another thing this means is that we can use QuickTime video effects anywhere we might use a decompressor, not only in connection with QuickTime movies. We can just as easily apply a transition between two arbitrary images (perhaps contained in two offscreen graphics worlds). I've seen this capability used in applications that support QuickTime video effects as transitions between QuickTime VR nodes. The default behavior of QuickTime VR is simply to jump from one node to the next. It's much nicer to render some video effect, say a nice smooth dissolve, when moving from node to node.

In this article and the next, we're going to work with QuickTime video effects. We'll see how to create the fire movie shown in **Figure 5** and how to apply a filter to a video track or image. We'll also see how to display and manage the effects parameters dialog box, which allows the user to select an effect and modify the parameters of that effect. Finally, we'll see how to apply an effect to only part of an existing movie and how to use effects as sources of sprite images.

Our sample application in these two articles is called QTEffects; its Test menu is shown in **Figure 7**.

Figure 7: *The Test menu of QTEffects.*

In this article, we'll see how to handle all these menu items except for the fourth (which happens to be grayed out) and the final two. We'll postpone consideration of those three items to our next article.

QuickTime Video Effects in Movies

It's extremely easy to add a video effect to a QuickTime movie. In the simplest case, where the effect lasts for the entire length of the movie, we just add an effects track to the movie. An *effects track* is a video track (of type VideoMediaType) whose media data is an effect description. An *effect description* is an atom container that indicates which effect to perform and which parameters, if any, to use when rendering the effect. The effect description also indicates which other tracks in the movie are to be used as the input sources for the effect. These are called the *effect source tracks* (or *effect sources*). A transition needs two source tracks; a filter needs one source track; a generator needs no source tracks. **Figure 8** illustrates the general structure of the fire movie shown in **Figure 5**.

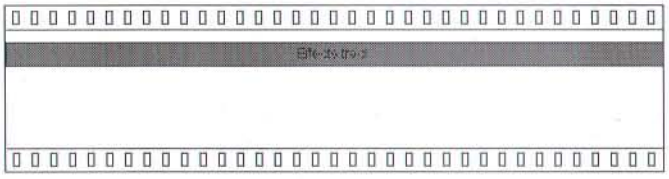


Figure 8: *The structure of a zero-source effect movie*

And **Figure 9** illustrates the general structure of a movie that contains a two-source effect (perhaps the zig-zag transition shown in **Figure 1**).

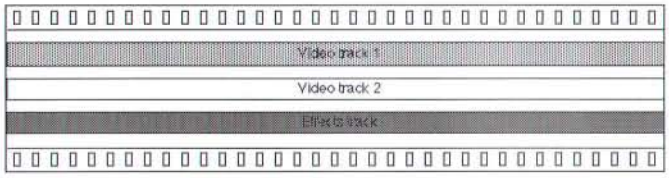


Figure 9: *The structure of a two-source effect movie*

The source tracks for a video effect can be any tracks that have the visual media characteristic, including video tracks, sprite tracks, text tracks, and others. In particular, because an effects track is a video track, it too can be a source track for another effects track. This allows us to *stack* effects, so that the output of one effect is used as input for another effect. For example, we could set up a cross-fade transition from one video track to another, and then apply a film noise filter to the resulting images. Keep in mind, however, that some effects can use a significant amount of CPU power, so that stacking effects may result in movies that do not play smoothly in real time on slower machines.

As we'll see in greater detail later, we need to connect an effects track to its source tracks by setting up track references from the effects track to the source tracks. These references tell QuickTime where to get the data for the effects track. We also need to configure the effects track's input map, so that the effects track knows how to interpret the data it receives from the source tracks. The source tracks operate as modifier tracks, whose data is not presented directly to the user; rather, their data is used as input for the effects track. This is important, particularly when we want to apply an effect to only part of a source track. You might think that we could just construct an effects track with the appropriate start time and duration, as shown in **Figure 10**.

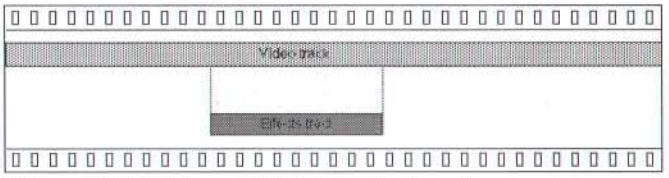


Figure 10: *A filter applied to part of a video track (wrong)*

But this won't work, since once we've created a track reference from the effects track to the video track and set the effects track's input map appropriately, the video track will send *all* of its data to the effects track, not just the data in the track segment that overlaps the effects track. To apply an effect to a part of a track, we can create another track that has the desired start time and duration and that references data in the video track. Then we use this new track segment as the source track for the effect, as shown in **Figure 11**. The new track segment doesn't contain a copy of the media data; instead, it contains references to the media data that already exists in the video track. So we don't increase the size of a movie file very much at all when we add effects to it.

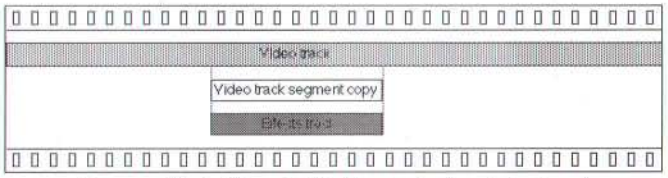


Figure 11: *A filter applied to part of a video track*

All three of the tracks shown in **Figure 11** are enabled; to prevent the original video track from covering up the effects track, we need to make sure that the effects track has a lower track layer than the video track. We'll see exactly how to do this in the next article, when we discuss applying effects to track segments.

It's worth mentioning that the QuickTime video effects architecture was originally designed to render effects in real time using software effects components (which, as we've seen, are image decompressor components). Recently,

Tired of Quicken®?

Appgen software runs on

Mac OS X

Linux and Windows

You now have the power to choose from this proven family of true accounting products. Visit www.appgen.com or call 1 800 231 0062 for more information about our unbeatable CPA Referral Program

APPGEN® Moneydance®

"As a user, Moneydance picks up where Quicken leaves off; as a CPA, Moneydance is an easier, more reliable, and safer way of maintaining your data."

—Peter J. Renzulli, CPA
Member AICPA

APPGEN® MyBooks®

True business accounting with audit trails for Small Office, Home Office and other small businesses with 1-5 users. And it's modifiable!

APPGEN® Accounting for Small Business

Fully customizable business accounting software for small businesses with up to hundreds of users

APPGEN® PowerWindows™ v5.1

The four-star rated software that forms the basis for all Appgen business products

from The Appgen Software Companies

QuickTime 5 added support for hardware acceleration of effects rendering. This acceleration is used only when the user's machine has the appropriate hardware installed, and it occurs automatically (without any intervention by the effects movie creator or the playback application).

It's also worth mentioning that a video effect can have more than two sources. QuickTime 5 introduced a three-source effect, the *traveling matte* effect. In these articles, we'll always work with two or fewer sources, but our code can in fact handle up to three.

Effects Utilities

Before we begin creating effects movies, let's take a brief moment to define a couple of functions that will be useful throughout our effects code.

Creating a Sample Description

When we build an effects track, we need to pass AddMediaSample an image description that provides information about the effect. In the past, we've always created sample descriptions and image descriptions by calling NewHandleClear and then setting the fields of the structure appropriately. When we are working with effects, however, we should use the function MakeImageDescriptionForEffect, which allocates a handle to an image description and fills in some of its fields; it also attaches an *image description extension* to the end of the image description. This extension indicates that that image description applies to an effect. For most purposes this extension is ignored, but it's necessary when we want to create stacked effects.

MakeImageDescriptionForEffect was introduced in QuickTime 4.0; if we want our code to run also under versions 3.x, we can set the USES_MAKE_IMAGE_DESC_FOR_EFFECT compiler flag to 0. Listing 1 shows our definition of EffectsUtils_MakeSampleDescription, which we'll call quite a few times in QTEffects to create an image description for an effect.

Listing 1: Creating a sample description for an effect

EffectsUtils_MakeSampleDescription

```
ImageDescriptionHandle EffectsUtils_MakeSampleDescription
      (OSType theEffectType, short theWidth, short
theHeight)
{
  ImageDescriptionHandle   mySampleDesc = NULL;

#if USES_MAKE_IMAGE_DESC_FOR_EFFECT
  OSErr                    myErr = noErr;

  // create a new sample description
  myErr = MakeImageDescriptionForEffect(theEffectType,
        &mySampleDesc);
  if (myErr != noErr)
    return(NULL);
#else
  // create a new sample description
  mySampleDesc = (ImageDescriptionHandle)
        NewHandleClear(sizeof(ImageDescription));
  if (mySampleDesc == NULL)
    return(NULL);

  // fill in the fields of the sample description
  (**mySampleDesc).cType = theEffectType;
  (**mySampleDesc).idSize = sizeof(ImageDescription);
  (**mySampleDesc).hRes = 72L << 16;
  (**mySampleDesc).vRes = 72L << 16;
  (**mySampleDesc).frameCount = 1;
  (**mySampleDesc).depth = 0;
  (**mySampleDesc).clutID = -1;
#endif

  (**mySampleDesc).vendor = kAppleManufacturer;
  (**mySampleDesc).temporalQuality = codecNormalQuality;
  (**mySampleDesc).spatialQuality = codecNormalQuality;
  (**mySampleDesc).width = theWidth;
  (**mySampleDesc).height = theHeight;

  return(mySampleDesc);
}
```

Notice that we need to set a few fields of the image description even if we call MakeImageDescriptionForEffect.

Creating an Effect Description

It's also useful to define a utility function to build an effect description. As we've learned, an effect description is an atom container that specifies an effect and its sources. Listing 2 shows the definition of our utility EffectsUtils_CreateEffectDescription. The essential step is to add an atom of type kParameterWhatName and ID kParameterWhatID whose data is the four-character code for the desired effect.

Listing 2: Creating an effect description

EffectsUtils_CreateEffectDescription

```
QTAtomContainer EffectsUtils_CreateEffectDescription
        (OSType theEffectType, OSType theSourceName1,
        OSType theSourceName2, OSType theSourceName3)
{
  QTAtomContainer   myEffectDesc = NULL;
  OSType            myType = EndianU32_NtoB(theEffectType);
  OSErr             myErr = noErr;

  // create a new, empty effect description
  myErr = QTNewAtomContainer(&myEffectDesc);
  if (myErr != noErr)
    goto bail;

  // create the effect ID atom
  myErr = QTInsertChild(myEffectDesc, kParentAtomIsContainer,
        kParameterWhatName, kParameterWhatID, 0,
        sizeof(myType), &myType, NULL);
  if (myErr != noErr)
    goto bail;

  // add the first source
  if (theSourceName1 != kSourceNoneName) {
    myType = EndianU32_NtoB(theSourceName1);
    myErr = QTInsertChild(myEffectDesc,
        kParentAtomIsContainer, kEffectSourceName, 1, 0,
        sizeof(myType), &myType, NULL);
    if (myErr != noErr)
      goto bail;
  }

  // add the second source
  if (theSourceName2 != kSourceNoneName) {
    myType = EndianU32_NtoB(theSourceName2);
    myErr = QTInsertChild(myEffectDesc,
        kParentAtomIsContainer, kEffectSourceName, 2, 0,
        sizeof(myType), &myType, NULL);
    if (myErr != noErr)
      goto bail;
  }
```

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."
– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."
– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"
– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."
– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."
– Joe Zobkiw, author of ***A Fragment of Your Imagination***

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: $256 (decimal)
Website price: $128 - $256
(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMÆSTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New in 2.0:

- **Very fast, HFS browser for viewing file tree of all volumes**
- **Extensibility for new Resorcerer Apprentices (CFM plug-ins)**
- **New AppleScript Dictionary ('aete') Apprentice Editor**
- **MacOS 8 Appearance Manager-savvy Control Editor**
- **PowerPlant text traits and menu command support**
- **Complete AIFF sound file disassembly template**
- **Big-, little-, and even mixed-endian template parsing**
- **Auto-backup during file saves; folder attribute editing**
- **Ships with PowerPC native, fat, and 68K versions**

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

```
    // add the third source
    if (theSourceName3 != kSourceNoneName) {
      myType = EndianU32_NtoB(theSourceName3);
      myErr = QTInsertChild(myEffectDesc,
          kParentAtomIsContainer, kEffectSourceName, 3, 0,
          sizeof(myType), &myType, NULL);
    }

bail:
    return(myEffectDesc);
}
```

EffectsUtils_CreateEffectDescription builds an effect description with up to three *source name atoms*, of type kEffectSourceName. The data in these atoms is a *source name*, of type OSType. Source names are used to link the source tracks to the effects track. These names are arbitrary, but Apple recommends using names of the form 'srcX', where *X* is an uppercase letter. In the file EffectsUtilities.h, we define these constants for our source names:

```
#define kSourceOneName              FOUR_CHAR_CODE('srcA')
#define kSourceTwoName              FOUR_CHAR_CODE('srcB')
#define kSourceThreeName            FOUR_CHAR_CODE('srcC')
#define kSourceNoneName               FOUR_CHAR_CODE('srcZ')
```

When we call EffectsUtils_CreateEffectDescription, we'll pass the constant kSourceNoneName for any unused sources.

Getting an Effect Type

Sometimes we might get our hands on an effect description and need to know what kind of effect it describes. We can get this information by inspecting the data of the atom of type kParameterWhatName and ID kParameterWhatID that's inside that effect description. The function EffectsUtils_GetTypeFromEffectDescription defined in Listing 3 accomplishes this.

Listing 3: Getting the type of an effect

EffectsUtils_GetTypeFromEffectDescription

```
OSErr EffectsUtils_GetTypeFromEffectDescription
      (QTAtomContainer theEffectDesc, OSType *theEffectType)
{
  QTAtom      myEffectAtom = 0;
  long        myEffectTypeSize = 0;
  Ptr         myEffectTypePtr = NULL;
  OSErr       myErr = noErr;

  if ((theEffectDesc == NULL) || (theEffectType == NULL))
    return(paramErr);

  myEffectAtom = QTFindChildByIndex(theEffectDesc,
        kParentAtomIsContainer, kParameterWhatName,
        kParameterWhatID, NULL);
  if (myEffectAtom != 0) {

    myErr = QTLockContainer(theEffectDesc);
    if (myErr != noErr)
      goto bail;

    myErr = QTGetAtomDataPtr(theEffectDesc, myEffectAtom,
        &myEffectTypeSize, &myEffectTypePtr);
    if (myErr != noErr)
      goto bail;

    if (myEffectTypeSize != sizeof(OSType)) {
      myErr = paramErr;
      goto bail;
    }

    *theEffectType = *(OSType *)myEffectTypePtr;
    *theEffectType = EndianU32_BtoN(*theEffectType);

    myErr = QTUnlockContainer(theEffectDesc);
  }

bail:
  return(myErr);
}
```

Notice that we call QTLockContainer on the effect description, even though it isn't strictly necessary here. As we learned in a previous article, QTGetAtomDataPtr returns a pointer to the actual leaf atom data. We need to call QTLockContainer only when we make calls that might move memory; in this case, we're just reading a few bytes into a local variable, and this operation will not cause any memory movement. The calls to QTLockContainer and QTUnlockContainer are fairly lightweight, so we'll make them anyway.

Generators

Let's begin our hands-on work with QuickTime video effects by building a movie that uses a generator, or zero-source effect. In this case, we'll build the fire movie shown earlier in Figure 5. This movie has only one track, which is an effects track and which has only one media sample. We'll set the dimensions of the effects track and its duration using some hard-coded values:

```
#define kDefaultTrackWidth        160
#define kDefaultTrackHeight         120
#define kEffectMovieDuration        (10 * kOneSecond)
```

We create the new movie file by calling CreateMovieFile, and then we create a new effects track and media like this:

```
myEffectTrack = NewMovieTrack(myMovie,
        IntToFixed(kDefaultTrackWidth),
        IntToFixed(kDefaultTrackHeight), kNoVolume);

myEffectMedia = NewTrackMedia(myEffectTrack, VideoMediaType,
        kOneSecond, NULL, 0);
```

Now we are ready to use the utility functions we defined in the previous section. We create the sample description and the effect description:

```
mySampleDesc = EffectsUtils_MakeSampleDescription
        (kFireCodecType, kDefaultTrackWidth,
        kDefaultTrackHeight);

myEffectDesc = EffectsUtils_CreateEffectDescription
        (kFireCodecType, kSourceNoneName, kSourceNoneName,
        kSourceNoneName);
```

The fire effect takes no sources, so we pass the constant kSourceNoneName for all three source name parameters.

Beta 3
now includes support for
Red Hat Linux
Sun Solaris

Middleware for the Mac

Need a Better Software Communication Solution?

We Built the Missing Piece

RapidMQ

Messaging & Queuing

RapidMQ is a flexible Messaging and Queuing solution enabling applications to communicate securely and reliably. RapidMQ helps solve software communication problems in many different markets.

Business Systems

- Communicate with customers, vendors, and partners using industry standard SSL security.
- Control the message traffic within your network through the use of Connections and Zones.
- Collect and analyze message traffic using RapidAdmin software.
- Quickly integrate RapidMQ using easy to understand programmatic interfaces.

E-Commerce

- Real-time data synchronization allows data to be cached in memory for fast access.
- Messaging infrastructure connects web applications to internal systems.
- Higher return on investment (ROI) gives you a better value than competing middleware solutions.
- Customer data is safely transferred using industry standard SSL security.

Consumer Software

- Automatically collect registration information from software even when the computer was not connected to the Internet at the time of installation.
- Allow users to purchase new features or updates directly through your software using industry standard SSL security.
- Provide product feedback and bug reporting interfaces within your application for direct delivery to your internal systems.

Mac

Jiiva, Inc.
www.jiiva.com/rapidmq

X
Built for Mac OS X

Now we are essentially done; we add the effect description as a media sample using the usual media-editing song-and-dance (BeginMediaEdits, AddMediaSample, EndMediaEdits, and InsertMediaIntoTrack). The key step is the call to AddMediaSample:

```
myErr = AddMediaSample(myEffectMedia, myEffectDesc, 0,
        GetHandleSize(myEffectDesc), kEffectMovieDuration,
        (SampleDescriptionHandle)mySampleDesc, 1, 0,
        &mySampleTime);
```

It's really just that easy to create a zero-source effects movie. Listing 4 shows the complete definition of QTEffects_MakeFireMovie, which we call in response to the "Make Fire Movie..." menu item.

Listing 4: Creating a zero-source effects movie

QTEffects_MakeFireMovie

```
void QTEffects_MakeFireMovie (void)
{
  FSSpec            myFile;
  Boolean           myIsSelected = false;
  Boolean           myIsReplacing = false;
  StringPtr         myPrompt =
  QTUtils_ConvertCToPascalString(kEffectsSaveMoviePrompt);
  StringPtr         myFileName =
  QTUtils_ConvertCToPascalString(kEffectsFireMovieFileName);
  Movie             myMovie = NULL;
  short             myMovieRefNum = kInvalidFileRefNum;
  short             myResID = movieInDataForkResID;
  Track             myEffectTrack = NULL;
  Media             myEffectMedia = NULL;
  QTAtomContainer   myEffectDesc = NULL;
  ImageDescriptionHandle
                    mySampleDesc = NULL;
  TimeValue         mySampleTime = 0;
  long              myFlags = createMovieFileDeleteCurFile
                        | createMovieFileDontCreateResFile;
  OSType            myType = FOUR_CHAR_CODE('none');
  OSErr             myErr = noErr;

  // ask the user for the name of the new movie file
  QTFrame_PutFile(myPrompt, myFileName, &myFile,
        &myIsSelected, &myIsReplacing);
  if (!myIsSelected)
    goto bail;          // deal with user cancelling

  // create a movie file for the destination movie
  myErr = CreateMovieFile(&myFile, sigMoviePlayer,
        smSystemScript, myFlags, &myMovieRefNum, &myMovie);
  if (myErr != noErr)
    goto bail;

  // select the "no controller" movie controller
  myType = EndianU32_NtoB(myType);
  SetUserDataItem(GetMovieUserData(myMovie), &myType,
        sizeof(myType), kUserDataMovieControllerType, 1);

  // create the effects track
  myEffectTrack = NewMovieTrack(myMovie,
        IntToFixed(kDefaultTrackWidth),
        IntToFixed(kDefaultTrackHeight), kNoVolume);
  if (myEffectTrack == NULL)
    goto bail;

  myEffectMedia = NewTrackMedia(myEffectTrack,
        VideoMediaType, kOneSecond, NULL, 0);
  if (myEffectMedia == NULL)
    goto bail;

  // create the sample description
  mySampleDesc = EffectsUtils_MakeSampleDescription
        (kFireCodecType, kDefaultTrackWidth,
        kDefaultTrackHeight);
  if (mySampleDesc == NULL)
    goto bail;

  // create the effect description
  myEffectDesc = EffectsUtils_CreateEffectDescription
        (kFireCodecType, kSourceNoneName, kSourceNoneName,
        kSourceNoneName);
  if (myEffectDesc == NULL)
    goto bail;

  // add the effect description as a sample to the effects track media
  myErr = BeginMediaEdits(myEffectMedia);
  if (myErr != noErr)
    goto bail;

  myErr = AddMediaSample(myEffectMedia, myEffectDesc, 0,
        GetHandleSize(myEffectDesc), kEffectMovieDuration,
        (SampleDescriptionHandle)mySampleDesc, 1, 0,
        &mySampleTime);
  if (myErr != noErr)
    goto bail;

  myErr = EndMediaEdits(myEffectMedia);
  if (myErr != noErr)
    goto bail;

  myErr = InsertMediaIntoTrack(myEffectTrack, 0,
        mySampleTime, kEffectMovieDuration, fixed1);
  if (myErr != noErr)
    goto bail;

  AddMovieResource(myMovie, myMovieRefNum, &myResID, NULL);

bail:
  if (myMovieRefNum != kInvalidFileRefNum)
    CloseMovieFile(myMovieRefNum);

  if (myMovie != NULL)
    DisposeMovie(myMovie);

  if (myEffectDesc != NULL)
    QTDisposeAtomContainer(myEffectDesc);

  if (mySampleDesc!= NULL)
    DisposeHandle((Handle)mySampleDesc);

  free(myPrompt);
  free(myFileName);

  return;
}
```

With the fire effect, the duration is fairly arbitrary. Any non-zero duration would produce the same visual output.

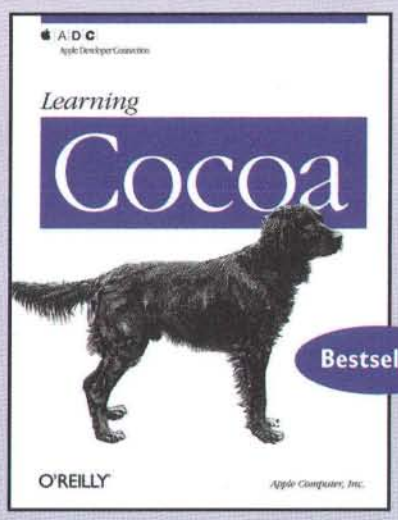
Filters

It's just about as easy to add a filter to a video track in an existing movie — for instance, to handle the "Add Film Noise To Movie" menu item. We add an effects track, whose media data consists of an effect description. This time, however, we need to specify a source name in the effect description. We'll call our utility EffectsUtils_CreateEffectDescription like this, passing kSourceOneName as the first source name parameter:

```
myEffectDesc = EffectsUtils_CreateEffectDescription
        (kFilmNoiseImageFilterType, kSourceOneName,
        kSourceNoneName, kSourceNoneName);
```

We also need to create an input map for the effects track, which specifies which track is to be used as the effect source. Listing 5 shows the code we use to create, configure, and set the input map.

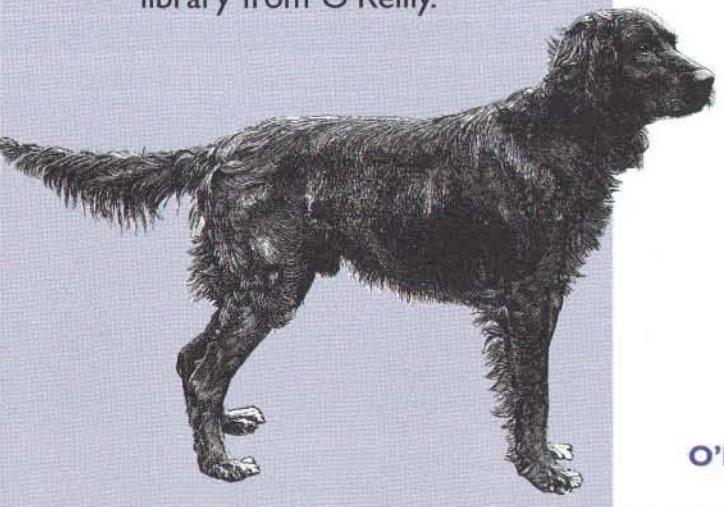
Mac® OS X rekindles the spark, O'Reilly fans the flame

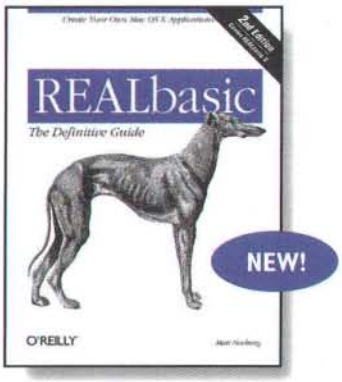


ISBN: 0-596-00160-6, $34.95

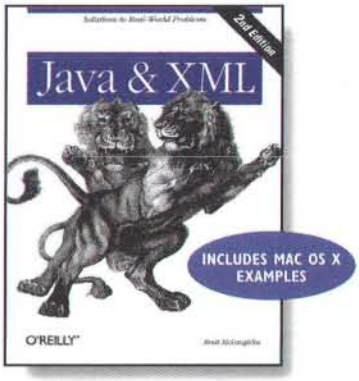
O'Reilly—your match to ignite the power of Mac OS X

Get ALL the benefits of Mac OS X —with a ready-made Mac OS X developer's library from O'Reilly.

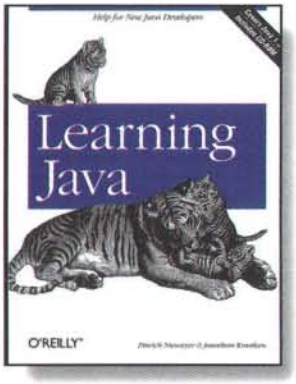




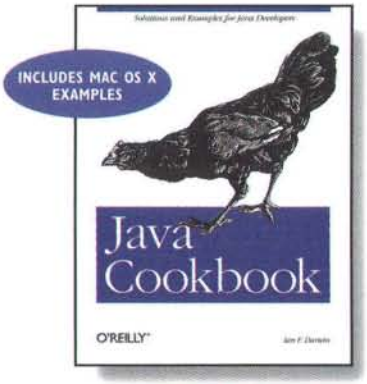
ISBN: 0-596-00177-0, $39.95



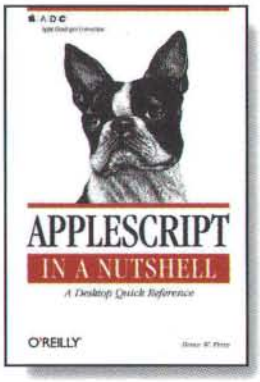
ISBN: 0-596-00197-5, $44.95



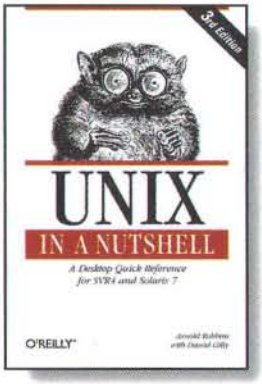
ISBN: 1-56592-718-4, $39.95



ISBN: 0-596-00170-3, $44.95



ISBN: 1-56592-841-5, $29.95



ISBN: 1-56592-427-4, $29.95

For more Mac OS X titles visit mac.oreilly.com
O'Reilly books are also available at your favorite bookstore.

800-998-9938
www.oreilly.com

O'REILLY®

Listing 5: Creating an input map for an effects track

QTEffects_AddFilmNoiseToMovie

```
// create the input map and add references for the first effects track
myErr = QTNewAtomContainer(&myInputMap);
if (myErr != noErr)
  goto bail;

myErr = EffectsUtils_AddTrackReferenceToInputMap
        (myInputMap, myTrack, mySrcTrack, kSourceOneName);
if (myErr != noErr)
  goto bail;

// add the input map to the effects track
myErr = SetMediaInputMap(myMedia, myInputMap);
```

An input map for an effects track is an atom container that holds one atom of type kTrackModifierInput for each source track that is sending data to the effects track. The ID of each such atom must be set to the reference index returned by AddTrackReference when the track reference between the effects track and that source track is created. Each atom of type kTrackModifierInput must contain at least two child atoms. One of these children is of type kTrackModifierType and specifies the kind of data the target track is going to receive from the source track; in the case of an effects track, the type of the modifier track input is kTrackModifierTypeImage. The second child atom in an input map entry atom specifies the name of the source track and is of type kEffectDataSourceType; the data in this atom is of type OSType. **Figure 12** shows the structure of the input map we'll use to add the film noise filter to a video track.

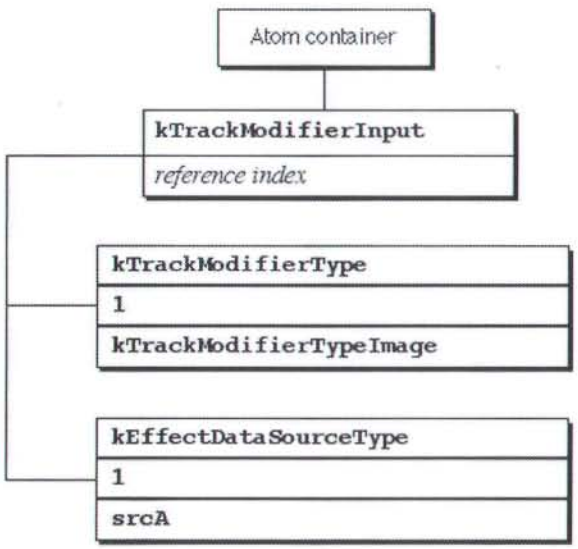


Figure 12: *The structure of an input map for an effects track*

Listing 6 shows our definition of the EffectsUtils_AddTrackReferenceToInputMap function, which we use to add the appropriate children to an existing input map for an effects track.

Listing 6: Adding track references to an input map

EffectsUtils_AddTrackReferenceToInputMap

```
OSErr EffectsUtils_AddTrackReferenceToInputMap
        (QTAtomContainer theInputMap, Track theTrack,
        Track theSrcTrack, OSType theSrcName)
{
  QTAtom        myInputAtom;
  long          myRefIndex;
  OSType        myType;
  OSErr         myErr = noErr;

  myErr = AddTrackReference(theTrack, theSrcTrack,
        kTrackReferenceModifier, &myRefIndex);
  if (myErr != noErr)
    goto bail;

  // add a reference atom to the input map
  myErr = QTInsertChild(theInputMap, kParentAtomIsContainer,
        kTrackModifierInput, myRefIndex, 0, 0, NULL,
        &myInputAtom);
  if (myErr != noErr)
    goto bail;

  // add two child atoms to the parent reference atom
  myType = EndianU32_NtoB(kTrackModifierTypeImage);
  myErr = QTInsertChild(theInputMap, myInputAtom,
        kTrackModifierType, 1, 0, sizeof(myType), &myType,
        NULL);
  if (myErr != noErr)
    goto bail;

  myType = EndianU32_NtoB(theSrcName);
  myErr = QTInsertChild(theInputMap, myInputAtom,
        kEffectDataSourceType, 1, 0, sizeof(myType),
&myType,
        NULL);

bail:
  return(myErr);
}
```

If EffectsUtils_AddTrackReferenceToInputMap seems familiar, that's because we've already bumped into similar functions (for instance, when we worked with sprite image overrides in "An Extremely Goofy Movie" in *MacTech*, April 2001).

Transitions

Adding a transition to a movie with two video tracks is really no more complicated than adding a filter to a movie with one video track. We pass kSourceTwoName as the second source name parameter when calling EffectsUtils_CreateEffectDescription, and we call EffectsUtils_AddTrackReferenceToInputMap a second time, to create a track reference between the effects track and the second source video track. For fun, let's see how to recreate our appearing-penguin movie using QuickTime video effects. We'll also take this opportunity to play a little more with our favorite Image Compression Manager functions GetMaxCompressionSize and CompressImage.

Given what we've learned so far, all we really need to do is create two video tracks to serve as the source tracks for a cross-fade transition. The first video track is an all-white frame that lasts for the duration of the movie; the second video track is the fully-opaque penguin picture, also lasting for the duration of the movie. As always, we'll set the duration of the movie to 10 seconds, this time using the constant kEffectMovieDuration. Then we'll add the effects track to the movie, specifying a cross-fade from the first source track to the second.

The two images that we'll use to create our video tracks are stored in our application's resource fork, in two 'PICT' resources with these IDs:

```
#define kWhiteRectID              129
#define kPenguinPictID            128
```

So we need to read each image and create a video track of the desired length. We'll split our work into two parts. First we'll write a utility, EffectsUtils_GetPictResourceAsGWorld, that reads a 'PICT' resource and draws it into an offscreen graphics world. Then we'll write another utility, EffectsUtils_AddVideoTrackFromGWorld, that creates a video track from the image in an offscreen graphics world. Once we've got these two utilities, we can create the two video tracks using the code shown in Listing 7.

Listing 7: Creating two video tracks from two 'PICT' resources

QTEffects_MakePenguinMovie

```
myErr = EffectsUtils_GetPictResourceAsGWorld(kWhiteRectID,
        kPenguinTrackWidth, kPenguinTrackHeight, 0, &myGW1);
if (myErr != noErr)
  goto bail;

myErr = EffectsUtils_GetPictResourceAsGWorld(kPenguinPictID,
        kPenguinTrackWidth, kPenguinTrackHeight, 0, &myGW2);
if (myErr != noErr)
  goto bail;

myErr = EffectsUtils_AddVideoTrackFromGWorld(&myMovie, myGW1,
        &mySrc1Track, 0, kEffectMovieDuration,
        kPenguinTrackWidth, kPenguinTrackHeight);
if (myErr != noErr)
  goto bail;

myErr = EffectsUtils_AddVideoTrackFromGWorld(&myMovie, myGW2,
        &mySrc2Track, 0, kEffectMovieDuration,
        kPenguinTrackWidth, kPenguinTrackHeight);
```

To create an offscreen graphics world that holds the image stored in a 'PICT' resource, we get the picture data from the resource (by calling GetPicture), create an offscreen graphics world of the required size (by calling QTNewGWorld), and then draw the picture data into that new graphics world (by calling DrawPicture). Before drawing into our graphics world, however, we need to call LockPixels to lock the offscreen pixel image. Listing 8 shows our definition of EffectsUtils_GetPictResourceAsGWorld.

Listing 8: Creating a graphics world from a 'PICT' resource

EffectsUtils_GetPictResourceAsGWorld

```
OSErr EffectsUtils_GetPictResourceAsGWorld (short theResID,
        short theWidth, short theHeight, short theDepth,
        GWorldPtr *theGW)
{
  PicHandle       myHandle = NULL;
  PixMapHandle    myPixMap = NULL;
  CGrafPtr        mySavedPort;
  GDHandle        mySavedDevice;
  Rect            myRect;
  OSErr           myErr = noErr;

  // get the current drawing environment
  GetGWorld(&mySavedPort, &mySavedDevice);

  // read the specified 'PICT' resource from the application's resource file
  myHandle = GetPicture(theResID);
  if (myHandle == NULL) {
    myErr = ResError();
    if (myErr == noErr)
      myErr = resNotFound;
    goto bail;
  }

  // set the size of the GWorld
  MacSetRect(&myRect, 0, 0, theWidth, theHeight);
```

```
  // allocate a new GWorld
  myErr = QTNewGWorld(theGW, theDepth, &myRect, NULL, NULL,
        kICMTempThenAppMemory);
  if (myErr != noErr)
    goto bail;

  SetGWorld(*theGW, NULL);

  // get a handle to the offscreen pixel image and lock it
  myPixMap = GetGWorldPixMap(*theGW);
  LockPixels(myPixMap);

  EraseRect(&myRect);
  DrawPicture(myHandle, &myRect);

  if (myPixMap != NULL)
    UnlockPixels(myPixMap);

bail:
  // restore the previous port and device
  SetGWorld(mySavedPort, mySavedDevice);

  if (myHandle != NULL)
    ReleaseResource((Handle)myHandle);

  return(myErr);
}
```

Now we want to create a video track in a movie that lasts for a specified duration and whose data is the image contained in an offscreen graphics world. Listing 9 shows the complete definition of EffectsUtils_AddVideoTrackFromGWorld. This function is a tad long, since we need to create a new track and add a media sample to it; we also need to call GetMaxCompressionSize and CompressImage to compress the data in the original graphics world to reduce the size of the resulting new movie track. Notice that we return the track identifier to the caller through the theSourceTrack parameter. (For more details on calling GetMaxCompressionSize and CompressImage, see "Making Movies" in *MacTech*, June 2000.)

Listing 9: Creating a video track from a graphics world

EffectsUtils_AddVideoTrackFromGWorld

```
OSErr EffectsUtils_AddVideoTrackFromGWorld (Movie *theMovie,
        GWorldPtr theGW, Track *theSourceTrack,
        long theStartTime, TimeValue theDuration,
        short theWidth, short theHeight)
{
  Media           myMedia;
  ImageDescriptionHandle
                  mySampleDesc = NULL;
  Rect            myRect;
  Rect            myRect2;
  Rect            myRect3;
  long            mySize;
  Handle          myData = NULL;
  Ptr             myDataPtr = NULL;
  GWorldPtr       myGWorld = NULL;
  CGrafPtr        mySavedPort = NULL;
  GDHandle        mySavedGDevice = NULL;
  PicHandle       myHandle = NULL;
  PixMapHandle    mySrcPixMap = NULL;
  PixMapHandle    myDstPixMap = NULL;
  OSErr           myErr = noErr;

  // get the current port and device
  GetGWorld(&mySavedPort, &mySavedGDevice);

  // create a video track in the movie
  *theSourceTrack = NewMovieTrack(*theMovie,
        IntToFixed(theWidth), IntToFixed(theHeight),
        kNoVolume);
  if (theSourceTrack == NULL)
    goto bail;

  myMedia = NewTrackMedia(*theSourceTrack, VideoMediaType,
        kVideoTrackTimeScale, NULL, 0);
  if (myMedia == NULL)
    goto bail;

  // get the rectangle for the movie
  GetMovieBox(*theMovie, &myRect);

  // begin editing the new track
  myErr = BeginMediaEdits(myMedia);
  if (myErr != noErr)
    goto bail;

  // create a new GWorld; we draw the picture into this GWorld and then compress it
  // (note that we are creating a picture with the maximum bit depth)
  myErr = NewGWorld(&myGWorld, 32, &myRect, NULL, NULL, 0L);
  if (myErr != noErr)
    goto bail;

  mySrcPixMap = GetGWorldPixMap(theGW);
  myDstPixMap = GetGWorldPixMap(myGWorld);
  LockPixels(myDstPixMap);

  // create a new image description; CompressImage will fill in the fields of this
structure
  mySampleDesc = (ImageDescriptionHandle)NewHandle(4);

  SetGWorld(myGWorld, NULL);
#if TARGET_OS_MAC
  GetPortBounds(theGW, &myRect2);
  GetPortBounds(myGWorld, &myRect3);
#endif
#if TARGET_OS_WIN32
  myRect2 = theGW->portRect;
  myRect3 = myGWorld->portRect;
#endif

  // copy the image from the specified GWorld into the new GWorld
  CopyBits((BitMapPtr)*mySrcPixMap, (BitMapPtr)*myDstPixMap,
        &myRect2, &myRect3, srcCopy, NULL);

  // restore the original port and device
  SetGWorld(mySavedPort, mySavedGDevice);

  myErr = GetMaxCompressionSize(myDstPixMap, &myRect, 0,
        codecNormalQuality, kJPEGCodecType, anyCodec,
        &mySize);
  if (myErr != noErr)
    goto bail;

  myData = NewHandle(mySize);
  if (myData == NULL)
    goto bail;

  HLockHi(myData);
#if TARGET_CPU_68K
  myDataPtr = StripAddress(*myData);
#else
  myDataPtr = *myData;
#endif
  myErr = CompressImage(myDstPixMap, &myRect,
        codecNormalQuality, kJPEGCodecType, mySampleDesc,
        myDataPtr);
  if (myErr != noErr)
    goto bail;

  myErr = AddMediaSample(myMedia, myData, 0,
        (**mySampleDesc).dataSize, theDuration,
        (SampleDescriptionHandle)mySampleDesc, 1, 0, NULL);
  if (myErr != noErr)
    goto bail;

  myErr = EndMediaEdits(myMedia);
  if (myErr != noErr)
    goto bail;
```

```
    myErr = InsertMediaIntoTrack(*theSourceTrack, theStartTime,
          0, GetMediaDuration(myMedia), fixed1);

bail:
    // restore the original port and device
    SetGWorld(mySavedPort, mySavedGDevice);

    if (myData != NULL) {
      HUnlock(myData);
      DisposeHandle(myData);
    }

    if (mySampleDesc!= NULL)
      DisposeHandle((Handle)mySampleDesc);

    if (myDstPixMap != NULL)
      UnlockPixels(myDstPixMap);

    if (myGWorld != NULL)
      DisposeGWorld(myGWorld);

    return(myErr);
}
```

See the file QTEffects.c for the complete listing of QTEffects_MakePenguinMovie, which we call in response to the "Make Fade-In Movie..." menu item. It's really just a longer version of QTEffects_MakeFireMovie (Listing 4) that incorporates the extra code in Listing 7.

Before we move on, it's worth reflecting on the fact that the movie created by QTEffects_MakePenguinMovie is now the *fourth* version of our penguin movie. We first created an appearing-penguin movie in "Making Movies" (cited earlier), where we built a video track with 100 frames, each frame having slightly more opacity than the previous. The total size of the movie file was about 470 kilobytes. In "A Goofy Movie" (March 2001), we created a second version of the penguin movie, using a sprite image in a key frame with zero opacity and 99 override frames that gradually increased the level of opacity of the sprite image. This version of the penguin movie file was only about 36 kilobytes. In the very next article ("An Extremely Goofy Movie", April 2001), we reworked that sprite version using a tween track to change the graphics mode of the penguin sprite image. The total size of that version was about 28 kilobytes. Finally, in this article, we've created a movie file that uses the cross-fade transition to blend from a totally white frame to the penguin image; this version is only about 10 kilobytes (most of which is occupied by the compressed penguin image).

One moral of this story is obvious: there is more than one way, using QuickTime, to skin a cat (or fade in a penguin). The first version uses a single video track. The second version uses a single sprite track. The third version uses a sprite track and a tween track. This fourth version uses two video tracks (the sources) and an effects track. None of these versions is inherently any better or worse than any of the others (though it's hard not to choke on the beefy size of the first version). Which of them we employ for a specific purpose depends on various factors. For instance, if we want the smallest file size, we would use the effects version; if we want to be able to add wiring to the movie, then a sprite version is preferable.

Effects Parameters

So far, our effect descriptions contain only a single kParameterWhatName atom and zero or more kEffectSourceName atoms. All of the built-in QuickTime video effects also support *effects parameters*, which specify additional information about the effect. For instance, the fire effect supports four parameters, which indicate the desired spread rate, sputter rate, water rate, and restart rate for the fire. The *sputter rate* (or *decay rate*) specifies how quickly the flames die down as they move upward. Larger values of the decay rate result in very low flames. (See Apple's effects documentation, cited at the end of this article, for descriptions of the other parameters.)

We specify a value for an effects parameter by inserting a parameter atom into the effect description. For instance, once we've created an effect description for the fire effect (by calling EffectsUtils_CreateEffectDescription), we can add a parameter atom to set the decay rate to 11, like this:

```
myRate = EndianS32_NtoB(11);
myErr = QTInsertChild(myEffectDesc, kParentAtomIsContainer,
        FOUR_CHAR_CODE('decy'), 1, 0, sizeof(myRate),
        &myRate, NULL);
```

The type of the parameter atom indicates the kind of parameter we are setting, and the data in the parameter atom is the desired value for that parameter.

Not all parameters are optional. With the SMPTE effects, the effect type indicates which of the four general classes of SMPTE effects (wipe, iris, radial, or matrix) the effect belongs to. To select a specific effect from those classes, we need to add a wipe ID parameter atom to the effect description. For instance, to specify the horizontal barn zig-zag effect (shown in Figure 1), we could execute this code:

```
myWipe = EndianS32_NtoB(kHorizontalBarnZigZagWipe);
myErr = QTInsertChild(myEffectDesc, kParentAtomIsContainer,
        FOUR_CHAR_CODE('wpID'), 1, 0, sizeof(myWipe),
        &myWipe, NULL);
```

The constant kHorizontalBarnZigZagWipe and others for the remaining SMPTE effects are defined in the file ImageCodec.h.

Using the Effects Parameters Dialog Box

When we build an effects movie, it would be nice to provide the user with an interactive way to set any of the optional effects parameters. To this end, the QuickTime video effects architecture includes support for displaying and managing the *effects parameters dialog box*, shown in Figure 13. (Sometimes this dialog box is also called the *standard parameters dialog box*.) As you can see, this dialog box includes a list of available effects (in this case, just the one-source effects) and some controls allowing the user to modify the parameters associated with the selected effect. It also includes a preview pane holding a poster image that is dynamically updated to reflect the current parameter settings.

PROGRAMMERS CREATE. SALESPEOPLE SELL. OR AT LEAST THAT'S THE WAY IT USED TO BE.

There's a revolution taking place. And you're in command.

With eSellerate—the revolutionary, turnkey e-commerce product from MindVision, creators of Installer VISE—you have the power to sell more software. Quickly *and* easily.

But that's only the beginning. Quite simply, eSellerate is not what you think. And it can do a whole lot more than you can imagine.

DEVELOPERS OF THE WORLD UNITE

Keep your current sales channels. With eSellerate, you're simply adding revolutionary sales capabilities. For the first time, users can purchase your software from within your application—providing them with instant use of the purchased software. Since all information is accessed "live" from the eSellerate network of servers at the time of purchase, your user will always receive the latest sales information, product version, and price. Everything else, from credit card processing, software downloading, installation, and serial number registration is handled seamlessly.

Best of all, it requires just one line of code added to your software.

esellerate.net

POWER TO THE PROGRAMMER

At its core, eSellerate is a library you simply link into your CodeWarrior or RealBasic project. It supports Mac OS 7.5 through OS X, 68K and PowerPC, Carbon and Mach-O. Just drop the library into your project and add one line of code to your source. That's all you need to do to put eSellerate to work for you—selling upgrades, converting trial users, and letting users purchase and install as a natural extension of your software.

Even after you link eSellerate into your software, you're in complete control. For starters, you decide when, where, and how the sales process takes place. You can continue to use your existing serial number scheme. What's more, you can change product prices, edit product information, upload new versions, and change product bundles. And, the changes are reflected immediately.

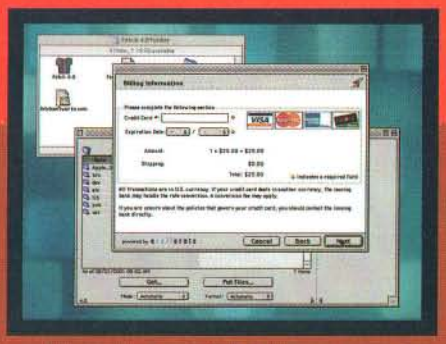
MINDVISION POWER TO THE PROGRAMMER

REVOLUTIONARY TECHNOLOGY

When a user decides to purchase your product, they automatically put an extremely sophisticated and robust process in motion.

First, eSellerate automatically establishes a secure connection with one of many servers, retrieving and offering the user your most up-to-date product information. Once the user enters their credit card information, it's securely encrypted and transmitted back for fraud screening and authorization. In just seconds, the credit card is authorized and all necessary registration information is securely sent to the user's machine where eSellerate decrypts the received data and completes the transaction (converting your trialware into a full version, for example).

From your user's perspective, the transaction is a seamless, instantaneous extension of your software. For you, it's an incredibly secure, dependable transaction you don't even have to think about. Utilizing extensive bandwidth, several redundancy measures, and the strongest exportable encryption, eSellerate keeps everything running smoothly—24 hours a day, seven days a week, 365 days a year.



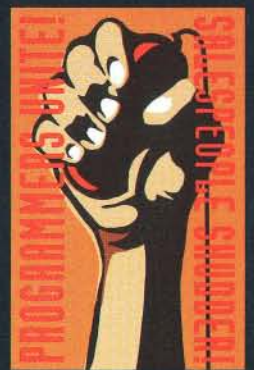
You control how and when your user is presented with the sales opportunity from within your software.

Start your revolution now with just one line of code.

LONG LIVE THE REVOLUTION

There are no setup costs or hidden fees. You pay only a small percentage on each sale—as low as 10%. And since eSellerate supplements your existing sales channels, instead of replacing them, there's absolutely no risk.

Try it right now. Just go to **www.esellerate.net/revolution** and set up a no-cost, no-risk, no-obligation publisher account. Then get ready to join the revolution.



SOFTWARE SALES REVOLUTION HAS BEGUN

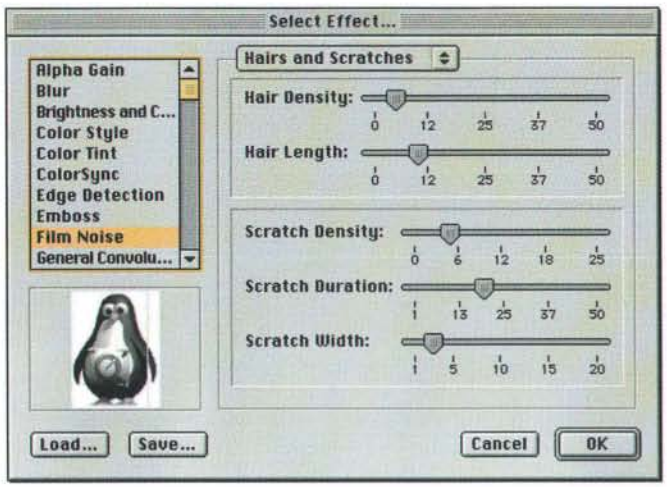


Figure 13: *The effect parameters dialog box/*

QuickTime provides the QTCreateStandardParameterDialog function for displaying the effects parameters dialog box, which is declared essentially like this:

```
OSErr QTCreateStandardParameterDialog (
        QTAtomContainer effectList,
        QTAtomContainer parameters,
        QTParameterDialogOptions dialogOptions,
        QTParameterDialog *createdDialog);
```

The effectList parameter specifies which effects we want to appear in the list on the left side of the dialog box. QuickTime also provides a function that we can use to get a list of all effects that take a certain number of sources:

```
myErr = QTGetEffectsList(&gEffectList, theSpecCount,
        theSpecCount, 0);
```

The second and third parameters to QTGetEffectsList specify the minimum and maximum number of sources; in this case, we set both of those parameters to the number of sources selected by the user. QTGetEffectsList returns, through its first parameter, an atom container that holds at least two atoms for every available effect that has the requisite number of sources. These two atoms specify the name and type of the effect. The atoms are sorted alphabetically by effect name. (That is, the atom of type kEffectNameAtom with ID 1 is the first name alphabetically; the atom of type kEffectNameAtom with ID 2 is next; and so forth.)

The second parameter to QTCreateStandardParameterDialog is an atom container in which information will be returned to us when the user finishes selecting an effect and its parameters. We need to allocate that atom container ourselves, like so:

```
myErr = QTNewAtomContainer(&gEffectDesc);

myErr = QTCreateStandardParameterDialog(gEffectList,
        gEffectDesc, 0, &gEffectsDialog);
```

The third parameter specifies some flags (which we set to 0 here) and the fourth parameter is the location of a variable of type QTParameterDialog; if QTCreateStandardParameterDialog completes successfully, it returns in that location an identifier for the effects parameters dialog box. We'll use that identifier in subsequent operations on the dialog box.

Setting the Poster Images

After we call QTCreateStandardParameterDialog, the effects parameters dialog box is not actually displayed on the screen until the dialog box receives an event. (As we'll see shortly, we pass events to the dialog box by calling QTIsStandardParameterDialogEvent.) This delay gives us an opportunity to do any necessary configuration in the dialog box before the user actually sees it. The main thing we want to do is set the poster image or images displayed in the box.

We set a poster image by calling the QTStandardParameterDialogDoAction function, which is declared essentially like this:

```
OSErr QTStandardParameterDialogDoAction (
        QTParameterDialog createdDialog, long action,
        void *params);
```

The action parameter specifies which action we want to perform on the dialog box. In QTEffects, we will use these three actions:

```
enum {
  pdActionConfirmDialog             = 1,
  pdActionSetPreviewPicture         = 6,
  pdActionModelessCallback        = 12
};
```

To set the preview image, we use the pdActionSetPreviewPicture action, in which case the params parameter is a pointer to a *parameter dialog box preview record*, declared like this:

```
struct QTParamPreviewRecord {
  long                  sourceID;
  PicHandle             sourcePicture;
};
```

The sourcePicture field contains a picture handle for the preview image, which must not be disposed until the dialog box is dismissed. The sourceID field indicates the index of the image. A filter should have one preview image with this field set to 1, and a transition should have two preview images with this field set to 1 and 2. Listing 10 shows how we would set the preview image for a filter.

Listing 10: Setting a preview image

QTEffects_DisplayDialogForSources

```
if (mySrcTrack != NULL) {
  gPosterA = GetTrackPict(mySrcTrack,
        GetMoviePosterTime(mySrcTrack));
  if (gPosterA != NULL) {
    QTParamPreviewRecord     myPreviewRecord;

    myPreviewRecord.sourcePicture = gPosterA;
    myPreviewRecord.sourceID = 1;
    myErr = QTStandardParameterDialogDoAction(gEffectsDialog,
        pdActionSetPreviewPicture, &myPreviewRecord);
  }
}
```

QuickTime provides a number of other selectors for customizing the effects parameters dialog box and its operation. For instance, to set a custom title on the dialog box, we can use the pdActionSetDialogTitle action selector, like this:

```
StringPtr myPtr = QTUtils_ConvertCToPascalString(kMyTitle);

myErr = QTStandardParameterDialogDoAction(gEffectsDialog,
        pdActionSetDialogTitle, myPtr);
free(myPtr);
```

See Apple's effects documentation for a complete list of the action selectors supported by QTStandardParameterDialogDoAction.

Handling Events in the Effects Parameters Dialog Box

Once we've configured the effects parameters dialog box to our liking, we need to start sending events to it so that it is displayed on the screen and the user can interact with it. We use the QTIsStandardParameterDialogEvent function to send events to that dialog box, like this:

```
myErr = QTIsStandardParameterDialogEvent(theEvent,
        gEffectsDialog);
```

QTIsStandardParameterDialogEvent determines whether the specified event is meant for the effects parameters dialog box (rather in the same way that IsDialogEvent determines whether an event is meant for a typical dialog box). If the event does apply to that dialog box, it's handled; in any case, QTIsStandardParameterDialogEvent returns a result code to its caller that indicates what action, if any, it took. We need to inspect that result code and react accordingly. Currently, QTIsStandardParameterDialogEvent returns one of four result codes:

- If codecParameterDialogConfirm is returned, the user has clicked the OK button; in this case, we need to tell QuickTime to fill the effect description we earlier passed to QTCreateStandardParameterDialog with atoms that reflect the user's selections in the dialog box. Then we should close the dialog box and use the information in that effects description.
- If userCanceledErr is returned, the user has clicked the Cancel button in the dialog box. In this case, we should close the dialog box and perform any necessary clean-up operations.
- If noErr is returned, the event was completely handled by the effects parameters dialog box code; we should proceed with further event processing.
- If featureUnsupported is returned, the event was not handled by the effects parameters dialog box code; we should allow the event to be processed by our application normally.

Listing 11 shows our definition of the QTEffects_HandleEffectsDialogEvents function, which we use to send events to the effect parameters dialog box and respond appropriately.

Listing 11: Handling dialog events

QTEffects_HandleEffectsDialogEvents

```
Boolean QTEffects_HandleEffectsDialogEvents
        (EventRecord *theEvent, DialogItemIndex theItemHit)
{
#pragma unused(theItemHit)
  Boolean     isHandled = false;
  OSErr       myErr = noErr;

  // pass the event to the standard effects parameters dialog box handler
  myErr = QTIsStandardParameterDialogEvent(theEvent,
        gEffectsDialog);

  // the result from QTIsStandardParameterDialogEvent tells us how to respond next
  switch (myErr) {

    case codecParameterDialogConfirm:
    case userCanceledErr:
      // the user clicked the OK or Cancel button;
      // dismiss the dialog box and respond accordingly
      gDoneWithDialog = true;

      if (myErr == codecParameterDialogConfirm)
        QTStandardParameterDialogDoAction(gEffectsDialog,
        pdActionConfirmDialog, NULL);
      QTDismissStandardParameterDialog(gEffectsDialog);
      gEffectsDialog = 0L;
      QTEffects_RespondToDialogSelection(myErr);
      isHandled = true;
      break;

    case noErr:
      // the event was completely handled by QTIsStandardParameterDialogEvent
      isHandled = true;
      break;

    case featureUnsupported:
      // the event was not handled by QTIsStandardParameterDialogEvent;
      // let the event be processed normally
      isHandled = false;
      break;

    default:
      // the event was not handled by QTIsStandardParameterDialogEvent;
      // do not let the event be processed normally
      isHandled = true;
      break;
  }

  return(isHandled);
}
```

Notice that the code for the codecParameterDialogConfirm result code calls QTStandardParameterDialogDoAction with the pdActionConfirmDialog action parameter; this fills in the effect description with the current values in the dialog box. That code also calls QTDismissStandardParameterDialog to close the dialog box and the application function QTEffects_RespondToDialogSelection to respond to the user's selection. We won't consider the QTEffects_RespondToDialogSelection function in this article, since it pretty much reprises code we've already seen to build an effects movie. That function does, however, contain some important clean-up code, shown in Listing 12.

Listing 12: Cleaning up after the dialog box is closed

QTEffects_RespondToDialogSelection

```
// standard parameter box has been dismissed; first do any necessary clean-up
gEffectsDialog = 0L;

// we're finished with the effect list and movie posters
if (gEffectList != NULL)
  QTDisposeAtomContainer(gEffectList);

if (gPosterA != NULL)
  KillPicture(gPosterA);

if (gPosterB != NULL)
  KillPicture(gPosterB);
```

Sending Events to the Effects Parameters Dialog Box

Now we know how to send events to the effects parameter dialog box and how to respond to the result codes that are returned to us. But when should we call QTEffects_HandleEffectsDialogEvents in our application code? On Macintosh systems, this is pretty easy, since our basic Macintosh application framework calls the function QTApp_HandleEvent for every event it receives from WaitNextEvent. Our application can inspect the gEffectsDialog global variable to see whether the effects parameter dialog box is currently displayed; if it is, we'll just call QTEffects_HandleEffectsDialogEvents, as shown in Listing 13.

Listing 13: Looking for events for the effects parameters dialog box

QTApp_HandleEvent

```
Boolean QTApp_HandleEvent (EventRecord *theEvent)
{
  Boolean     isHandled = false;

  // see if the event is meant for the effects parameter dialog box
  if (gEffectsDialog != 0L)
    isHandled = QTEffects_HandleEffectsDialogEvents(theEvent,
        0);

  return(isHandled);
}
```

On Windows, things are a bit trickier here. In our Windows application framework, QTApp_HandleEvent is called only when a movie window is open. So we can't rely on QTApp_HandleEvent to trigger the QTEffects_HandleEffectsDialogEvents function. Instead, we can use SetModelessDialogCallbackProc to install a callback function to handle Windows messages that apply to the effects parameters dialog box. (This function works equally well with modal dialog boxes, so don't worry about the name.) We'll call SetModelessDialogCallbackProc like this:

```
SetModelessDialogCallbackProc(FrontWindow(),
(QTModelessCallbackUPP)QTEffects_EffectsDialogCallback);
```

The specified callback procedure, QTEffects_EffectsDialogCallback, is called by QTML when it's handling events in dialog boxes. When our callback function is executed, QTML has already done any control tracking for controls in the dialog box. If a control has been selected, its ID is passed to us in the theItemHit parameter. Listing 14 shows our definition of QTEffects_EffectsDialogCallback.

Listing 14: Handling events for the effects parameters dialog box

QTEffects_EffectsDialogCallback

```
static void QTEffects_EffectsDialogCallback
        (EventRecord *theEvent, DialogRef theDialog,
```

```
        DialogItemIndex theItemHit)
{
  QTParamDialogEventRecord myRecord;

  myRecord.theEvent = theEvent;
  myRecord.whichDialog = theDialog;
  myRecord.itemHit = theItemHit;

  if (gEffectsDialog != 0L) {
    QTStandardParameterDialogDoAction(gEffectsDialog,
        pdActionModelessCallback, &myRecord);

    // see if the event is meant for the effects parameters dialog box
    QTEffects_HandleEffectsDialogEvents(theEvent,
        theItemHit);
  }
}
```

As you can see, we pass the event to QTEffects_HandleEffectsDialogEvents. (We also pass the index of the item hit, but it's ignored by that function.) We also call QTStandardParameterDialogDoAction, this time with the action pdActionModelessCallback. This is some magic that ensures that QTML properly updates the dialog box and its controls.

One last "gotcha" on Windows: we need to make sure that idle events are sent to the dialog box, so that it can run the effect in the preview pane. To accomplish this, we attach a custom window procedure to the dialog box by calling QTMLSetWindowWndProc, like this:

```
QTMLSetWindowWndProc(FrontWindow(),
        QTEffects_CustomDialogWndProc);
```

QTEffects_CustomDialogWndProc, defined in Listing 15, is called whenever the dialog box receives a message

Listing 15: Handling messages for the effects parameters dialog box

QTEffects_CustomDialogWndProc

```
LRESULT CALLBACK QTEffects_CustomDialogWndProc (HWND theWnd,
        UINT theMessage, UINT wParam, LONG lParam)
{
  EventRecord     myEvent = {0};

  if (!gDoneWithDialog && (theMessage == 0x7FFF))
    QTEffects_EffectsDialogCallback(&myEvent,
        GetNativeWindowPort(theWnd), 0);

  return(DefWindowProc(theWnd, theMessage, wParam, lParam));
}
```

As you can see, QTEffects_CustomDialogWndProc looks for messages of the type 0x7FFF (which is a special message produced by QTML to simulate Macintosh idle events); when it finds one, and if the dialog box is still active, it calls the function QTEffects_EffectsDialogCallback with an event record for an idle event.

Effects Parameter Files

Notice that the effects parameters dialog box in Figure 13 contains two buttons, labeled "Save..." and "Load...". These buttons allow the user to save the effects parameters currently displayed in the dialog box and to reload a saved set of parameters. For various purposes, it might be useful to perform these actions programmatically. For instance, once the user has selected a set of parameters for an effect, our application might want to save them into a file, whence we can retrieve them the next time the application is run. The format of these files is publicly defined and is indeed quite easy to read and write.

An *effects parameter file* is a file that specifies an effect and zero or more of its parameters; it may also specify the poster picture that appears in the effects parameters dialog box. An effects parameter file is organized as a series of "classic" atoms. Currently three kinds of atoms are included in one of these files:

- An atom of type 'qtfx' (required). The atom data is an atom container that holds information about the effect type and parameters. In other words, the atom data is an effect description.

- An atom of type 'pnot' (optional). The atom data is organized as a preview resource record (of type PreviewResourceRecord). This atom specifies the type and index of some other atom, which contains the actual poster data. Usually the other atom is of type 'PICT'.

- An atom of type 'PICT' (optional). The atom data is a picture that's used as the poster image in the effects parameters dialog box.

Other atoms may be included in an effects parameter file; applications that aren't expecting other atoms should be smart enough to skip them. By convention, an effects parameter file has the file extension '.qfx'; on Macintosh systems, the file type is 'qtfx'.

Currently, QuickTime does not provide any functions for reading or writing effects parameter files, but based on what we've learned hitherto (especially in "The Atomic Café" in *MacTech*, September 2000), we can easily write our own. Listing 16 shows a simple routine that we can use to open an effects parameter file and read the data of the 'qtfx' atom it contains.

Listing 16: Getting an effect description from an effects parameter file

EffectsUtils_GetEffectDescFromQFXFile

```
QTAtomContainer EffectsUtils_GetEffectDescFromQFXFile
        (FSSpec *theFSSpec)
{
  Handle      myEffectDesc = NULL;
  short       myRefNum = 0;
  long        mySize = 0L;
  OSType      myType = 0L;
  long        myAtomHeader[2];
  OSErr       myErr = noErr;

  myErr = FSpOpenDF(theFSSpec, fsRdPerm, &myRefNum);
  if (myErr != noErr)
    goto bail;

  SetFPos(myRefNum, fsFromStart, 0);

  while ((myErr == noErr) && (myEffectDesc == NULL)) {
    // read the atom header at the current file position
    mySize = sizeof(myAtomHeader);
    myErr = FSRead(myRefNum, &mySize, myAtomHeader);
    if (myErr != noErr)
      goto bail;

    mySize = EndianU32_BtoN(myAtomHeader[0]) -
                                        sizeof(myAtomHeader);
    myType = EndianU32_BtoN(myAtomHeader[1]);

    if (myType == FOUR_CHAR_CODE('qtfx')) {
      myEffectDesc = NewHandleClear(mySize);
      if (myEffectDesc == NULL)
        goto bail;

      myErr = FSRead(myRefNum, &mySize, *myEffectDesc);

    } else {
      SetFPos(myRefNum, fsFromMark, mySize);
    }
  }

bail:
  return((QTAtomContainer)myEffectDesc);
}
```

The effect description returned by this function can be used anywhere we use an effect description.

Conclusion

In this article, we've seen how to create movies that contain QuickTime video effects. We've worked with generators, filters, and transitions, and we've seen how to display and manage the effects parameters dialog box. We've also seen how to read data from an effects parameter file. The QuickTime video effects architecture provides a rich source of new capabilities that we can tap into with some very simple programming. The only really new thing we've encountered in this article is building effect descriptions, and even that turns out to be just another exercise in building atom containers.

You already know what's in store for us in the next article: we're going to see how to apply effects to images (not just to movie tracks). We're also going to see how to apply an effect to part of a movie, and how to use an effect as the image for a sprite. At some point in the distant future, we'll even learn how to write our own custom effects.

Acknowledgements and References

Many thanks are due to Tom Dowdy for reviewing this article and suggesting a number of improvements. Tom also wrote the code snippet in the *Letter from the Ice Floe*, Dispatch 24 (found at http://developer.apple.com/quicktime/icefloe/dispatch024.html) on which Listing 16 is based. The Apple documentation for the QuickTime video effects architecture can be found at http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmEffects.htm.

PROGRAMMER'S CHALLENGE

By Bob Boonstra, Westford, MA

NASSI-SCHNEIDERMAN

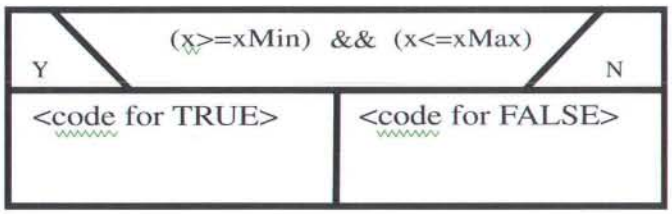
Perhaps you remember Nassi-Schneiderman diagrams as an alternative to traditional flow charts. Or perhaps you remember them as Chapin Charts. Whether you remember them or not, it's time to dig out those old computer science textbooks and bone up, because this month we're going to be drawing some of them.

There is no prototype for this Challenge, because you are going to build a complete Macintosh application. The basic requirement is that you open a text file containing valid C/C++ source code and generate a Nassi-Schneiderman diagram for each routine included in that file. To do that, you will need to provide menu options to open a file, allowing the user to navigate through the file system to select a file. You should parse the source code and open a window for each routine, drawing in the window the diagram that describes the program logic. The user must be allowed to move and resize the windows, and you should draw the diagram at a level of detail that fits into the window as sized by the user. You must provide a means for the user to select a section of the diagram, or click on a section of interest, and zoom in on that section. The zoomed-in display should increase the level of detail displayed, until no further increase in detail is possible. Your code must allow the user to switch to another application, and must properly update the window contents when appropriate.

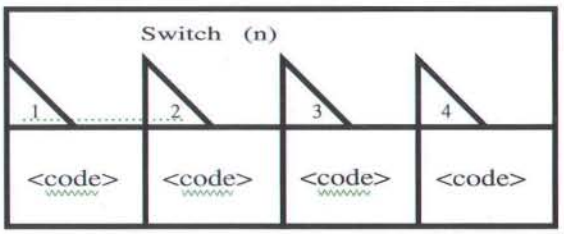
A Nassi-Schneiderman diagram consists of four simple block types: an instruction block, an alternative (e.g., if-then-else) block, a multiple choice (e.g., switch) block, and an iteration (e.g., do ... while) block.

The instruction block is simply a block with code inside:

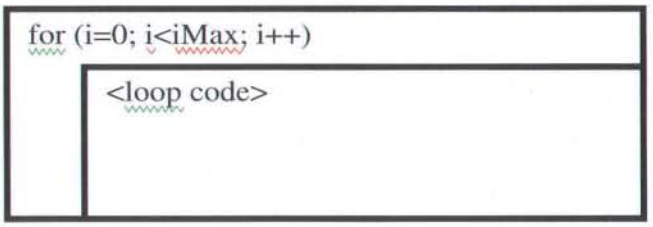
The alternative block looks like this:



The multiple choice block looks like this:



And the loop block:



Normally, these diagrams would contain pseudo-code that describes the intent of the actual code. Obviously, we won't have pseudo-code available, so you'll have to display something based on the code itself. Exactly what you display is at your discretion, but it should provide enough information so that you can identify the code associated with a block in the source file, subject to screen real estate limitations.

In creating the diagrams, you are encouraged to provide the user with additional features. For example, you might allow the user to change the text font and text size, and then adjust the text in the diagram if more or less text fits inside the boxes. You might provide a Window menu with the name of each diagrammed routine. You might provide an option to bring up the full text of a block inside a window. You can provide keyboard shortcuts or use modifier keys to implement special options. You might add a magnification capability that adds scroll bars to the window (in addition to the required zoom feature). You can add preference settings that make sense for your application. There might be a useful way to use color.

In the event the code contains a construct that cannot reasonably be diagramed (e.g., a goto statement), you should highlight the associated block in the diagram in some way and treat the offending construct as a no-op.

This will be a native PowerPC Challenge, using the latest version of CodeWarrior, or the development environment of your choice (provided I have a copy – email progchallenge@mactech.com to check before you use something else). You can develop for Mac OS 9 or Mac OS X. Evaluation of your entry will be subjective, based first on the

required features, then on optional features, and finally on general usability and look-and-feel. To ensure that I fairly evaluate your solution, you should include in your submission a list of any optional features you incorporated.

Three Months Ago Winner

After a Challenge absence of more than three years, **Jeff Mallett** (Boulder Creek, CA) returns to take first place in the June Dots Challenge. The object of this Challenge was to win a round-robin tournament of the game Dots (or Dots and Boxes). Dots is played on an NxN grid where players take turns connecting adjacent dots horizontally and vertically to enclose boxes. The player capturing the most boxes wins the game. The Challenge was actually scored based on minimizing the number of boxes captured by the opposing player, incorporating the usual efficiency requirement by adding a penalty of 1% for each millisecond of execution time. Solutions were also required to display the game state and the current score after each move.

Jeff maintains four doubly-linked lists of boxes in gList, one list for boxes with 0 edges (0-boxes), another for boxes with 1 edge (1-boxes), etc. He also maintains a data structure called a 2-path (TwoPathRecordType), which is a sequence of connected boxes each of which has two or three existing edges (2-boxes or 3-boxes, respectively).

The heavy lifting is done in the ComputerTurn routine, and I'll try to describe the logic. If there are no 3-boxes, the code looks to see if all unfilled boxes are 2-boxes. If so, any move is going to give a box to the opponent, so Jeff picks the move that gives the minimum away to the opponent. Otherwise, he selects a 0-box, or (as a second choice) a 1-box, provided it is "safe", where an "unsafe" box is one for which adding an edge creates a 3-box.

If there are 3-boxes, and there are safe places to move next, Jeff takes the 3-box and plays again. If the open edge of the 3-box is at the board boundary, he takes the box. Otherwise, he takes as many 3-boxes as he can while saving the moves that give the opponent a square ("handouts"). If a move that gives a handout captures half or more of the remaining boxes, Jeff makes that move. And finally, if forced, he makes a move that gives the opponent the smallest sequence of boxes.

As the comments in Jeff's code indicate, his solution is actually based on 15-year-old code, translated from Pascal into C/C++ for the Challenge. Jeff mentions that the time penalty in the problem caused him to significantly "dumb down" the program, removing enough of the look-ahead code to make it a fast, if mediocre, player.

The second-place entry, from Greg Sadetsky, is based (with permission) on a JavaScript program by UCLA Professor Thomas S. Ferguson. In addition to providing some very entertaining commentary that I wish we had the space to publish, Greg included the URL for the JavaScript code (http://www.stat.ucla.edu/~tom/Games/dots&boxes.html), as well as a page of links to other analyses of the game (http://dmoz.org/Games/Paper_and_Pencil/Dots_and_Boxes/).

The table below lists, for each of the solutions submitted, the number of cells captured by each solution in the tournament, the number of cells captured by the opposing player, the execution time in milliseconds, and the score earned by each solution (with lower scores being better). The table also includes the code and data size for each solution, and the programming language used. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

Name	Player Cells	Opponent Cells	Time (msec)	Score	Code	Data	Lang
Jeff Mallett (94)	3218	1362	833.7	2061.5	11572	908	C++
Greg Sadetsky (14)	3128	1452	1566.6	2753.8	17192	1936	C
Ernst Munter (751)	2529	2010	721.4	3009.9	10684	586	C++
Tom Saxton (185)	1914	2666	3262.5	8777.5	7220	581	C++
Randy Boring (142)	668	3422	53118.1	145423.8	17676	498	C
T. R.	1752	2297	367.1	2962.6	5188	307	C++

Top Contestants ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points (24 mo)	Wins (24 mo)	Total Points
1.	Munter, Ernst	271	10	758
2.	Rieken, Willeke	73	3	134
3.	Saxton, Tom	71	2	189
4.	Taylor, Jonathan	56	2	56
5.	Wihlborg, Claes	49	2	49
6.	Shearer, Rob	48	1	62
7.	Maurer, Sebastian	38	1	108
10.	Mallett, Jeff	20	1	114
11.	Truskier, Peter	20	1	20

... and the Top Contestants Looking for a Recent Win

In order to give some recognition to other participants in the Challenge, we also list the high scores for contestants who have accumulated points without taking first place in a Challenge during the past two years. Listed here are all of those contestants who have accumulated 6 or more points during the past two years.

Rank	Name	Points (24 mo)	Total Points
8.	Boring, Randy	32	144
9.	Sadetsky, Gregory	22	24
12.	Schotsman, Jan	14	14
13.	Nepsund, Ronald	10	57
14.	Day, Mark	10	30
15.	Jones, Dennis	10	22
16.	Downs, Andrew	10	12
17.	Desch, Noah	10	10
18.	Duga, Brady	10	10
19.	Fazekas, Miklos	10	10
20.	Flowers, Sue	10	10
21.	Strout, Joe	10	10
22.	Nicolle, Ludovic	7	55
23.	Hala, Ladislav	7	7
24.	Leshner, Will	7	7
25.	Miller, Mike	7	7
26.	Widyatama, Yudhi	7	7
27.	Heithcock, JG	6	43

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points
2nd place	10 points
3rd place	7 points
4th place	4 points
5th place	2 points
finding bug	2 points
suggesting Challenge	2 points

Here is Jeff's winning Boxes solution:

Boxes.cpp
Copyright © 1986-2001
Jeff Mallett

```
// Dots & Boxes
//
// Copyright 1986-2001 Jeff Mallett. All rights reserved.
// For licensing contact jeffm@zillions-of-games.com
//
// Written in Lightspeed Pascal for the Mac
//   Last revision in Pascal: Oct 14, 1986
// Ported from Pascal to C/C++ and adapted for contest: May-June 2001
//   (It uses inline/new/delete/template/etc., but it's really more C
//   than C++ since there are no classes. If I had more time I would
//   have converted it to be object-oriented as well.)
//
// Programmer's Challenge Entry
// This version does not play Dots & Boxes completely naively — e.g.
//   it understands the sacrificing boxes to keep the initiative — but
//   neither does it play very skillfully. This is because the
//   challenge awards points based on boxes and time taken, not
//   on who actually wins the games. As penalties are awarded
//   for every millisecond of thinking, a quick mediocre player is
//   likely to gain more points than a slow master. Therefore I
//   dumbed down the program considerably by ripping out code that
//   determined how to correctly play networks of 1-boxes
//   connected with 2-paths of length 1 or 2. Now the program
//   doesn't have a clue what to do with unsafe 1-boxes and never
//   sacrifices boxes early in the game, but it also saves a
//   lot of processing time.
//

#define DRAWING
#define EDGECOUNT_FIELD
//#define COORDINATE_FIELDS

#define TICKSEED LMGetTicks()
#define ASSERT assert

#include <assert.h>
#include <stdlib.h>
#include <limits.h>

#include "Dots.h"

#include <LowMem.h>
```

ENUMS

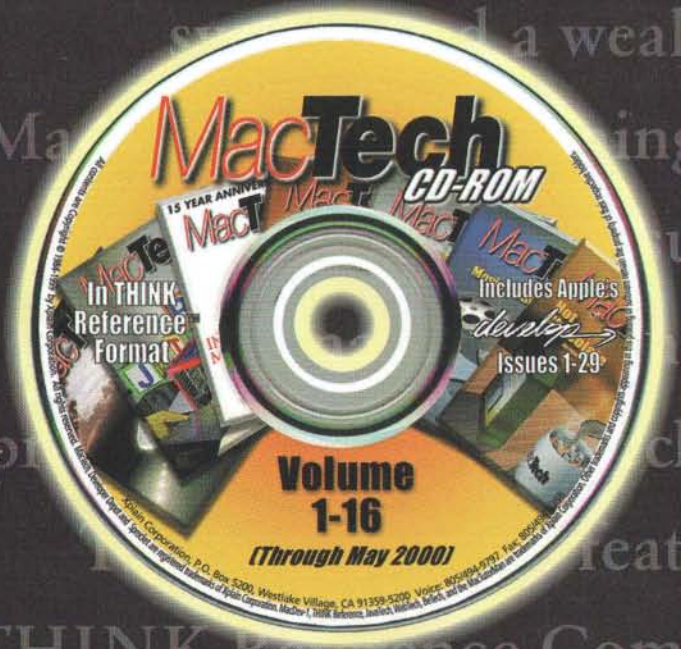
```
enum direction {left=0, up, right, down};
enum who {firstplayer=0, secondplayer};
enum pathdirection {behind=0, ahead};
// BORDER_VALUE: box is off the board
// MARK_VALUE: when calculating a path, a box with this value is already on it
enum boxflag {BORDER_VALUE=-2, MARK_VALUE=-1, NO_VALUE=0};
// NOT_LOOP: 2-path has two ends
// ALMOST_LOOP: 2-path's ends terminate at the same 0-box or 1-box
// LOOP: 2-path has no ends
```

THE EASY, COMPACT WAY TO SOLVE PROBLEMS!

MacTech® CD-ROM

www.mactech.com

The MacTech CD-ROM with THINK Reference is the essential reference resource for Macintosh programmers. This CD includes the THINK Reference personal database system and a wealth of Macintosh programming databases, featuring almost 200 issues of the journal of Macintosh programming – MacTech Magazine. This release also features the THINK Reference Compiler, which allows you to compile HTML files into your own compact, searchable THINK Reference databases.



PO Box 5200 • Westlake Village, CA • 91359-5200
800/MACDEV-1 (800/622-3381) • Outside US/Canada: 805/494-9797
Fax: 805/494-9798 • E-mail: orders@devdepot.com

```
enum loopvalue {NOT_LOOP=0, ALMOST_LOOP, LOOP};
```

TYPEDEFs

```
typedef struct BoxRecordType* PtrToBoxRecord;
typedef struct TwoPathRecordType* PtrToTwoPathRecord;

struct TwoPathRecordType {
  int pathsize;
  loopvalue loop;
  PtrToBoxRecord path;
  PtrToTwoPathRecord previous, next;
};

struct BoxRecordType {
#ifdef COORDINATE_FIELDS
  short xcoord, ycoord;        // Location of box
#endif
#ifdef EDGECOUNT_FIELD
  short edgecount;             // Number of edges
#endif
  Boolean edges[4];        // Is there an edge in this direction?
  PtrToBoxRecord previous, next;
                                  // previous & next elements in
gList[]
  boxflag flag;                // Used for marking borders and searched boxes
// 2-path info
  PtrToTwoPathRecord twopath;  // the 2-path this is in
  PtrToBoxRecord pnext[2];     // the two connecting boxes in the 2-path
};

struct MoveRecordType {
  PtrToBoxRecord box;
  direction dir;
};
```

GLOBALS

```
PtrToBoxRecord gBoxes;   // Array holding all the boxes
int gArraySizeY;         // Array width of gBoxes
int gBoardSizeX, gBoardSizeY;// Size of the board in boxes across and
down
PtrToBoxRecord gList[4]; // Pointers to doubly-linked lists of n-edge boxes
PtrToBoxRecord gListBookmark[2];
        // Next box on gList[n] to look at for a safe move
int gOffset[4];          // Offset in gBoxes for going this direction

PtrToTwoPathRecord gTwoPaths[3];// Pointers to a doubly-linked lists of 2-
paths
                    // [0]:length > 2  [1]:length 1  [2]:length 2
Boolean gPathsComputed;      // Have the 2-paths ever been recorded in
gTwoPaths?

Boolean gSafe[2];
          // Is there a move on an n-box that doesn't give the opponent a 3-box?
Boolean gSafetyCheckNeeded;
          // Need to check whether position has become dangerous?
PtrToBoxRecord gLastSafeBox[2];//The most recent box found safe on list n

int gScore[2];           // gScore[w] is the # of boxes player w has
completed
int gTotalScore;         //The sum of the two player's score
int gMaxScore;           //The maximum possible value of gTotalScore
who gCurrentPlayer;        //The current player
```

MACROS

```
#define EVEN(x)          (((x) & 1) == 0)
#define ODD(x)             (((x) & 1) != 0)

#define EDGE(x,y,d)        gBoxes[x*gArraySizeY + y].edges[d]
#define BOX(x,y)           (&gBoxes[x*gArraySizeY + y])

#define FOR_LIST(pList, whichList)  for (pList =
gList[whichList]; pList; pList = pList->next)

#define FOR_LIST2(pList, whichList, start) \
  if (gList[whichList]) { \
    if (!start || NumberOfEdges(start) != whichList) \
      start = gList[whichList]; \
    pList = start; do {

#define END_FOR_LIST2(pList, whichList, start) \
      pList = pList->next; \
      if (!pList) pList = gList[whichList]; \
    } while (pList != start); \
  }

#define FOR_EACH_OPEN_DIRECTION(box) \
{ direction f_dir = left; const Boolean *pEdges = (box)-
>edges; \
  do { \
    if ( !*(pEdges + (f_dir)) ) { \
      PtrToBoxRecord f_box = Go(box, f_dir);

#define END_FOR_EACH_OPEN_DIRECTION \
    } \
  } while ((f_dir = (direction)(f_dir+1)) <= down); \
}

#define  FOR_PATHDIRECTION(pdir)    pdir = behind; do
#define  END_FOR_PATHDIRECTION(pdir)  while ((pdir =
(pathdirection)(pdir+1)) <= ahead);
```

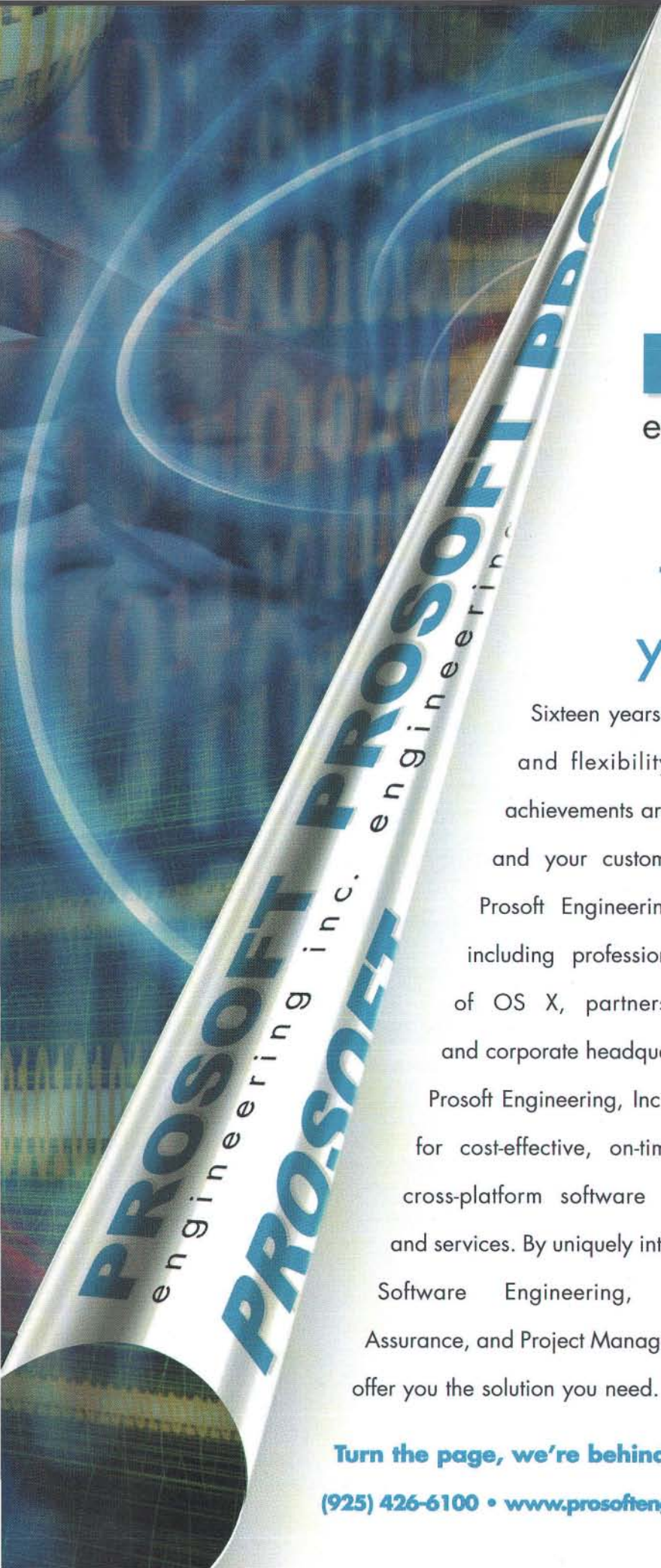
PROTOTYPES

```
#ifdef DRAWING
extern void DrawInitial(int x, int y, who person);
extern void DrawScore(who w);
extern void DrawMove(int x, int y, enum direction d);
extern void InitialDrawing();
#endif

void Initialize();

void CheckSafety(int listindex);
void Clear(PtrToBoxRecord box);
void ComputeAPath(PtrToBoxRecord posinedgelist,
PtrToTwoPathRecord existingtwopath);
void ComputePaths();
void UpdatePathFrom(PtrToBoxRecord box);
void AddEdge(PtrToBoxRecord box, direction d, who whoseturn);
int ExpectedScore(PtrToTwoPathRecord ignoreThis);
int CountTinyTwoPaths( );
int CountSurroundingPathSizes(PtrToBoxRecord box);
PtrToTwoPathRecord MinimumTwoPath();
void HandOut(int hotype, MoveRecordType &move);
direction HardHeartedHandout(PtrToBoxRecord box);
Boolean FindSafeMoveOnList(int listindex, MoveRecordType
&move);
void ComputerTurn(MoveRecordType &move);
Boolean MakeRealMove(MoveRecordType move, who whoseturn);
void ConvertFromDotLine(MoveRecordType &move, const DotLine
&dotline);
void ConvertToDotLine(MoveRecordType move, DotLine &dotline);
```

INLINES

```
// Returns X coordinate value for the box
inline int GetX (PtrToBoxRecord box) {
#ifdef COORDINATE_FIELDS
  return box->xcoord;
#else
  return (box - gBoxes) / gArraySizeY;
#endif
}

// Returns Y coordinate value for the box
inline int GetY (PtrToBoxRecord box) {
#ifdef COORDINATE_FIELDS
  return box->ycoord;
#else
  int dindex = box - gBoxes;
  int xcoord = dindex / gArraySizeY;
  return dindex - xcoord * gArraySizeY;
#endif
}

// Returns the box in direction dir
inline PtrToBoxRecord Go (PtrToBoxRecord box, direction dir)
{
  return box + gOffset[dir];
}
```

```
// Returns the opposite path direction to the given path direction
inline pathdirection OppositePathDirection (pathdirection
pathdir) {
  return (pathdirection) (1 - pathdir);
}

// Returns the direction opposite to the given direction
inline direction OppositeDirection (direction dir) {
  return (direction) ((dir+2)%4);
}

// Returns the next direction in a clockwise fashion
inline direction NextDirection (direction dir) {
  return (dir == down) ? left : (direction) (dir + 1);
}

// Goes to the next player
inline void NextPlayer (who &whoseturn) {
  whoseturn = (who) (1 - whoseturn);
}

// Produces a random number between 0 and n-1 inclusive
inline int rnd (int n) {
  return ((long)rand() * n) / ((long)RAND_MAX + 1);
}

// Returns a direction chosen at random (or at least different)
inline direction RandomDirection () {
  return (direction) (rand() & 3);
}

// Given a 3-box on a 2-path, returns the path direction out of the 3-box
inline pathdirection OutPathDirection (PtrToBoxRecord box) {
  return box->pnext[ahead] ? ahead : behind;
}

// Returns true iff the box is within the boundaries.
inline Boolean IsInsideBoard (PtrToBoxRecord box) {
  return box->flag != BORDER_VALUE;
}

// Returns a direction randomly such that !box->edges[d]
inline direction GetOpenDirection (PtrToBoxRecord box) {
  // cycle through directions starting at random direction dir
  direction d = RandomDirection();
  const Boolean *p = box->edges;
  while (*(p+d))
    d = NextDirection(d);
  return d;
}

// Sets up a move from box in a random direction with no edge
inline void PlayAny (PtrToBoxRecord box, MoveRecordType
&move) {
  move.box = box;
  move.dir = GetOpenDirection(box);
}

// Plays a move on the path p
inline void PlayMoveOnPath(PtrToTwoPathRecord p,
MoveRecordType &move) {
  move.box = p->path;
  move.dir = (p->pathsize == 2) ?
        HardHeartedHandout(move.box) :
GetOpenDirection(move.box);
}

// Returns whether or not boxes a and b are connected
inline Boolean AreConnected(PtrToBoxRecord a, PtrToBoxRecord
b) {
  FOR_EACH_OPEN_DIRECTION(a)
  {
    if (f_box == b)
      return true;
  } END_FOR_EACH_OPEN_DIRECTION;
  return false;
}
```



PROSOFT
engineering inc.

Look beyond technology and you will find us.

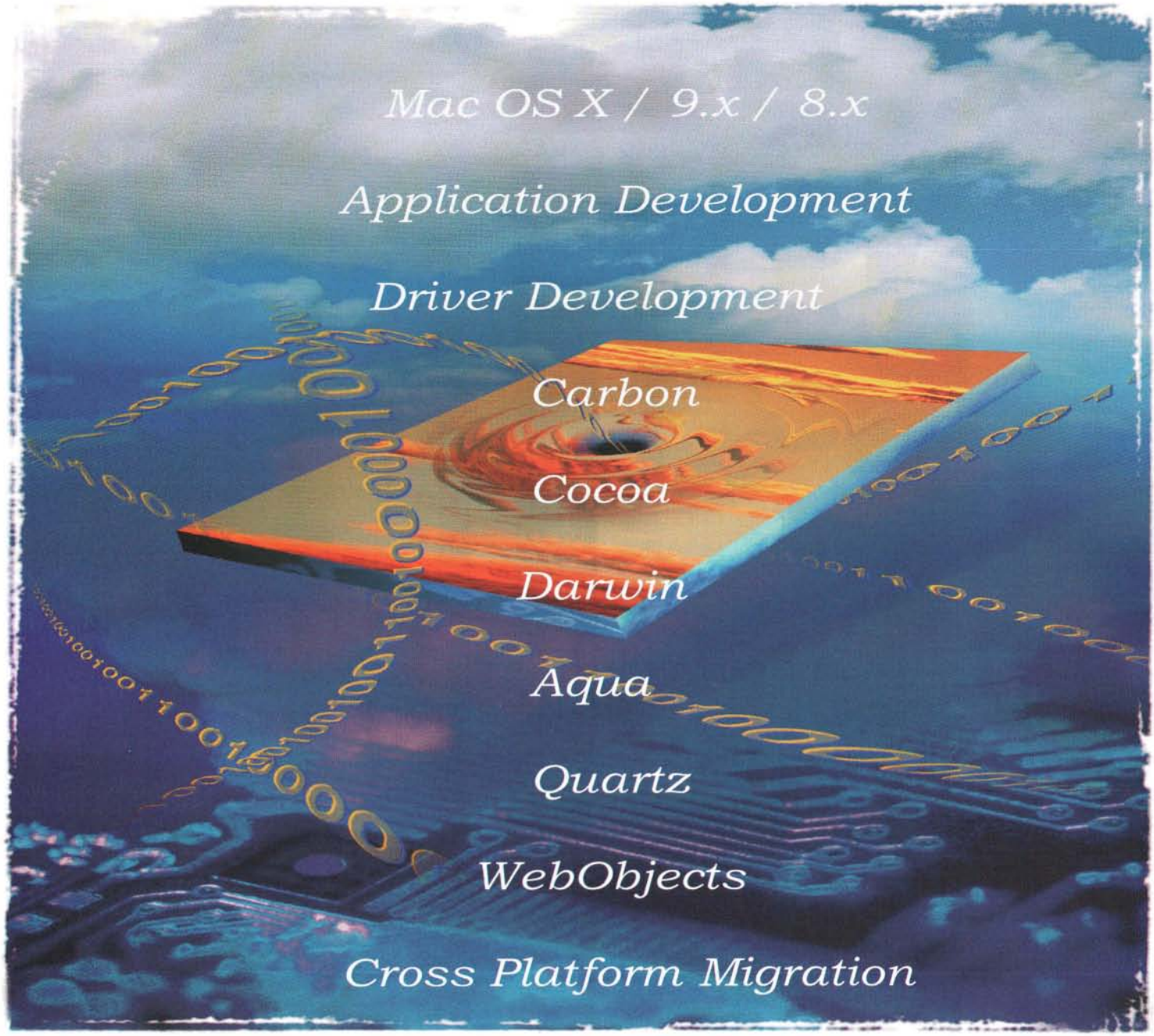
Sixteen years of experience coupled with the resources and flexibility you need. It's no surprise that our achievements are integrated into many of the products you and your customers have come to rely on every day. Prosoft Engineering, Inc. offers unsurpassed advantages including professional engineers with in-depth knowledge of OS X, partnerships with the top industry leaders, and corporate headquarters located in the heart of Silicon Valley.

Prosoft Engineering, Inc., is your source for cost-effective, on-time, custom cross-platform software solutions and services. By uniquely integrating Software Engineering, Quality Assurance, and Project Management, we offer you the solution you need.

X
Built for Mac OS X

Turn the page, we're behind your future.

(925) 426-6100 • www.prosofteng.com • info@prosofteng.com



We have the best engineers around. If it's Mac, we've done it.

- 15 Year Track Record of Delivering Consistent Customer Satisfaction
- Highly Skilled Quality Assurance & Testing Team
- Development Centers Nationwide
- Over 175 Engineers
- Extensive Configuration Testing Facilities

VANTEON™
Delivering Innovative Engineering Solutions

For more information please call 1-800-266-5046, e-mail at mac@vanteon.com or visit us at www.vanteon.com

a new kind of software development.

on time.

Without sufficient resources or the right expertise, your project is at risk of shipping late. Atimi guarantees reliable and timely Macintosh software development. You get to market on schedule—and to the satisfaction of both you and your customer. With a wide array of technical expertise, Atimi offers a full range of services:

Classic Mac OS to Mac OS X application porting (Carbon and Cocoa) | Windows to Mac OS application porting | Application development | Driver development (printer, PostScript, USB, Firewire, etc.) | Networking development (wireless and wireline)

Mac

Contact us now and learn more about how Atimi can help you with your project.

Call 604.813.1319 or visit us at www.atimi.com.

Atimi

Software development. On-time.

```
// Returns the number of edges of the box
inline int NumberOfEdges (PtrToBoxRecord box) {
#ifdef EDGECOUNT_FIELD
  return box->edgecount;
#else
  int count = 0;
  Boolean *p = box->edges;
  if (*(p++)) ++count;
  if (*(p++)) ++count;
  if (*(p++)) ++count;
  if (*p)     ++count;
  return count;
#endif
}
```

LIST TEMPLATES

```
// Adds the element to the beginning of the given list
template<class T> void AddToList(T element, T &list) {
  element->previous = NULL;
  element->next = list;
  list = element;
  if (element->next)
    element->next->previous = element;
}

// Removes the element from the given list
template<class T> void RemoveFromList(T element, T &list) {
  if (element->previous)
    element->previous->next = element->next;
  else
    list = element->next;
  if (element->next)
    element->next->previous = element->previous;
}

// Inserts toinsert into the list after the onlist box
// Note: toinsert must not be on a list currently
template<class T> void InsertAfter(T toinsert, T onlist) {
  if (onlist->next)
    onlist->next->previous = toinsert;
  toinsert->next = onlist->next;
  onlist->next = toinsert;
  toinsert->previous = onlist;
}

// Adds path to the front of the appropriate gTwoPaths list
inline void AddToPathsList(PtrToTwoPathRecord path) {
  AddToList(path, gTwoPaths[path->pathsize <= 2 ? path-
>pathsize : 0]);
}

// Removes path from the appropriate gTwoPaths list
inline void RemoveFromPathsList(PtrToTwoPathRecord path) {
  RemoveFromList(path, gTwoPaths[path->pathsize <= 2 ? path-
>pathsize : 0]);
}
```

Initialize

```
// Sets up variables for a game.
void Initialize()
{
  int i, j;
  int arraySizeX, arrayElements;

  srand(TICKSEED);

  arraySizeX = gBoardSizeX + 2; // borders on each side
  gArraySizeY = gBoardSizeY + 2;
  arrayElements = arraySizeX * gArraySizeY;
  gBoxes = new BoxRecordType [arrayElements];
  ASSERT(gBoxes);

  gOffset[left]  = -gArraySizeY;
  gOffset[up]    = -1;
  gOffset[right] = gArraySizeY;
  gOffset[down]  = 1;

  PtrToBoxRecord p = gBoxes;
  for (i = 0; i < arraySizeX; i++)
    for (j = 0; j < gArraySizeY; j++)
    {
      p->edges[left]  = p->edges[up] = p->edges[right] =
                p->edges[down] = false;
      p->flag = BORDER_VALUE;
      p->pnext[ahead] = p->pnext[behind] = NULL;
      p->twopath = NULL;
#ifdef COORDINATE_FIELDS
      p->xcoord = i; p->ycoord = j;
#endif
#ifdef EDGECOUNT_FIELD
      p->edgecount = 0;
#endif
      ++p;
    }

  gList[0] = BOX(1,1);
  gList[0]->previous = NULL;

  for (i = 1; i <= 3; i ++)
    gList[i] = NULL;

  PtrToBoxRecord last = NULL;
  for (i = 1; i <= gBoardSizeX; i++)
  {
    p = BOX(i, 1);
    for(j = 1; j <= gBoardSizeY; j++)
    {
      p->flag = NO_VALUE;
      if (last)
      {
        last->next = p;
        p->previous = last;
      }
      last = p;
      ++p;
    }
  }
  last->next = NULL;

  // Mix up list elements randomly
    for (i = gBoardSizeX; i >= 1; i --)
    for (j = 1; j <= gBoardSizeY; j ++)
    {
      int i2 = rnd(gBoardSizeX)+1;
      int j2 = rnd(gBoardSizeY)+1;
      if (i!=i2 || j!=j2)
      {
        RemoveFromList(BOX(i, j), gList[0]);
        InsertAfter(BOX(i, j), BOX(i2, j2));
      }
    }
  ASSERT(!gList[0]->previous);

  gSafe[0] = gSafe[1] = true;
  gPathsComputed = gSafetyCheckNeeded = false;
  gMaxScore = gBoardSizeX * gBoardSizeY;
  gTotalScore = gScore[firstplayer] = gScore[secondplayer] =
0;
  gListBookmark[0] = gListBookmark[1] =
    gLastSafeBox[0] = gLastSafeBox[1] = NULL;
}   // Initialize
```

ENGINE

CheckSafety

```
// Re-evaluates gSafe
// If we can't add an edge to a 0-box or 1-box without changing
// a 2-box into a 3-box set gSafe[listindex]=false
void CheckSafety(int listindex) {

  // Optimization: Check the box that was found safe last time
  if (gLastSafeBox[listindex])
  {
    if (NumberOfEdges(gLastSafeBox[listindex]) <= 1)
    {
      FOR_EACH_OPEN_DIRECTION(gLastSafeBox[listindex])
      {
        if (NumberOfEdges(f_box) != 2)
          return; // safe
      } END_FOR_EACH_OPEN_DIRECTION;
    }
    gLastSafeBox[listindex] = NULL;
  }
```

```
  // Check 0 and 1 lists
  PtrToBoxRecord box;
  FOR_LIST(box, listindex)
  {       // Check a box on list
    FOR_EACH_OPEN_DIRECTION(box)
    {
      if (NumberOfEdges(f_box) != 2)
      {
        gLastSafeBox[listindex] = box;
        return;
      }
    } END_FOR_EACH_OPEN_DIRECTION;
  }

  gSafe[listindex] = false; // unsafe
}   // CheckSafety
```

Clear

```
// Initializes/Reinitializes the path data for a single record
void Clear (PtrToBoxRecord box) {

  PtrToTwoPathRecord p = box->twopath;
  if (p)
  {
    RemoveFromPathsList(p);
    if (!--p->pathsize)
    {  // Box is only thing on twopath
      delete p;
    } else
    {
      AddToPathsList(p);
      if (p->path == box)
      {
        ASSERT(box->pnext[ahead]);
        p->path = box->pnext[ahead];
      }
      pathdirection pdir = OutPathDirection(box);
      box->pnext[pdir]->pnext[OppositePathDirection(pdir)] =
NULL;
    }
    box->twopath = NULL;
    box->pnext[ahead] = box->pnext[behind] = NULL;
  }
}   // Clear
```

ComputeAPath

```
// Given a 2-box or a 3-box, computes the path from that box
void ComputeAPath (PtrToBoxRecord posinedgelist,
PtrToTwoPathRecord existingtwopath) {

  Boolean pathend;
  PtrToBoxRecord temp;
  pathdirection pdir;
  int pathlength = -1;      // set to -1 because posinedge is counted twice
  PtrToBoxRecord end[2], beyondend[2];
  beyondend[ahead] = beyondend[behind] = NULL;
  loopvalue loop = NOT_LOOP;
  PtrToTwoPathRecord twopath;

  if (existingtwopath)
  {
    twopath = existingtwopath;
    RemoveFromPathsList(twopath);
  } else
  {
    twopath = new TwoPathRecordType;
    ASSERT(twopath);
  }

  FOR_PATHDIRECTION(pdir)
  {       // each iteration moves to the end of the path in the behind/ahead
direction
    temp = posinedgelist;
    do {    // each iteration marks one more square in the path
      pathlength++;
      temp->flag = MARK_VALUE;
      temp->twopath = twopath;
      pathend = true;
      // check for a continuation of the path in the pdir direction
      FOR_EACH_OPEN_DIRECTION(temp)
      {  // each iteration checks one direction for an adjoining 2/3-square
        // don't need to check if f_box inside board because borders have
        // BORDER_VALUE
        if (!f_box->flag)
        {       // found square next to current square that hasn't been marked
          int n = NumberOfEdges(f_box);
          if (n >= 2)
            {
            // include this square in path
            temp->pnext[pdir] = f_box;
            f_box->pnext[OppositePathDirection(pdir)] =
temp;
            temp = f_box;
            if (n == 2)
              pathend = false;
            else // n == 3
            {   // f_box is on path so add it, but don't look further
              pathlength++;
              temp->flag = MARK_VALUE;
              temp->twopath = twopath;
            }
            break;
          }
          // n < 2
          if ( pdir == ahead || temp != posinedgelist )
          {
            beyondend[pdir] = f_box;
            break;
          }
        }
      } END_FOR_EACH_OPEN_DIRECTION;
      // Either all directions have been looked at or a path continuation has
      // been found
    } while (!pathend);
    temp->pnext[pdir] = NULL;
    end[pdir] = temp;
  } END_FOR_PATHDIRECTION(pdir);

  // Clear flags
  for (temp = end[ahead]; temp; temp = temp->pnext[behind])
    temp->flag = NO_VALUE;

  // Is a loop or near loop?
  if (pathlength > 2)
  {
    if (beyondend[ahead] && beyondend[ahead] ==
beyondend[behind])
      loop = ALMOST_LOOP;
    else if (AreConnected(end[ahead], end[behind]))
      loop = LOOP;
  }

  twopath->pathsize = pathlength;
  twopath->loop = loop;
  twopath->path = end[behind];
  AddToPathsList(twopath);

}   // ComputeAPath
```

ComputePaths

```
// Finds all 2-paths and stores them by forming a doubly-linked list for each
path
void ComputePaths() {

  PtrToBoxRecord posinedgelist;

  FOR_LIST(posinedgelist, 2)
  {
    if (!posinedgelist->twopath)
      ComputeAPath(posinedgelist, NULL);
          // Compute path containing this element
  }
  gPathsComputed = true;
}   // ComputePaths
```

UpdatePathFrom

```
// Updates the paths that the newly changed box was in if needed
void UpdatePathFrom (PtrToBoxRecord box) {

  int count;

  // Update path lists
```

```
  count = NumberOfEdges(box);
  if (count == 4)
    Clear(box);
  else if (count == 2)
  {
    // Could be added to a 2-path
    PtrToTwoPathRecord twopath = NULL;
    FOR_EACH_OPEN_DIRECTION(box)
    {
      if (f_box->twopath)
        if (!twopath)
          twopath = f_box->twopath; // Recalculate this 2-path
        else if (f_box->twopath != twopath)
        {  // Combining two 2-paths
          RemoveFromPathsList(f_box->twopath);
          break;
        }
    } END_FOR_EACH_OPEN_DIRECTION;
    ComputeAPath(box, twopath);
  }
  else if (count == 3 &&
      !(box->twopath && box->twopath->pathsize == 1))
        // don't need to update a length 1 path
    ComputeAPath(box, box->twopath);
}     // UpdatePathFrom
```

AddEdge

```
// Adds an edge to box at x,y and direction d. If the edge completes a square
// update score, etc.
// If safe might need to be changed then checkneeded will be set to true.
void AddEdge (PtrToBoxRecord box, direction d, who whoseturn)
{

  box->edges[d] = true;
#ifdef EDGECOUNT_FIELD
  ++box->edgecount;
#endif

  if (!IsInsideBoard(box))
    return;

  int count = NumberOfEdges(box);
  if (count == 4)
  {
    gScore[whoseturn]++;
#ifdef DRAWING
    DrawScore(whoseturn);
    DrawInitial(GetX(box), GetY(box), whoseturn);
#endif
    ++gTotalScore;
    RemoveFromList(box, gList[3]); // Take box off of old edge list
    // Don't bother putting it on gList[4]
  }
  else
  {
    RemoveFromList(box, gList[count-1]); // Take box off of old edge
list
    AddToList(box, gList[count]); // Add box to the beginning of a new
list

    if (count == 2)
      gSafetyCheckNeeded = true;
  }
}   // AddEdge
```

ExpectedScore

```
// Return estimate of the number of boxes we'll capture
int ExpectedScore(PtrToTwoPathRecord ignoreThis) {

  ASSERT(gPathsComputed);

  int expected, tinycount1, tinycount2, loopcount[3];

  loopcount[NOT_LOOP] = loopcount[ALMOST_LOOP] =
        loopcount[LOOP] = tinycount1 = tinycount2 =
expected = 0;

  // Count

  PtrToTwoPathRecord p;
  for (p = gTwoPaths[1]; p; p = p->next)
    ++tinycount1;
  for (p = gTwoPaths[2]; p; p = p->next)
    ++tinycount2;
  for (p = gTwoPaths[0]; p; p = p->next)
  {
    expected += p->pathsize; // Add boxes for 2-paths
    ++loopcount[p->loop];
  }

  // Subtract out ignoreThis
  if (ignoreThis->pathsize == 1)
    -tinycount1;
  else if (ignoreThis->pathsize == 2)
    -tinycount2;
  else {
    expected -= ignoreThis->pathsize;
    -loopcount[ignoreThis->loop];
  }

  if (loopcount[NOT_LOOP] || loopcount[ALMOST_LOOP] ||
                                    loopcount[LOOP])
  {
    // Subtract handouts
    expected -= loopcount[NOT_LOOP] * 2;
    expected -= loopcount[ALMOST_LOOP] * 3;
    expected -= loopcount[LOOP] * 4;

    // but disregard final handout
    expected += loopcount[NOT_LOOP] ? 2 : 4;
  }

  // Add boxes for tiny 2-paths
  expected += tinycount1/2 + 2 * (tinycount2/2);
        // Get half of each rounded down
  if (ODD(tinycount1) && ODD(tinycount2))
    expected += 2; // Get last [2], e.g. 1 [1] and 1 [2]

  return expected;
}   // ExpectedScore
```

CountTinyTwoPaths

```
// Counts length=1 2-paths and length=2 2-paths (no 3's).
int CountTinyTwoPaths ( ) {

  int count = 0;

  PtrToTwoPathRecord p;
  for (p = gTwoPaths[1]; p; p = p->next)
    if (NumberOfEdges(p->path) == 2)
      ++count;
  for (p = gTwoPaths[2]; p; p = p->next)
    if (NumberOfEdges(p->path) == 2 &&
        NumberOfEdges(p->path->pnext[ahead]) == 2)
      ++count;

  return count;
}   // CountTinyTwoPaths
```

CountSurroundingPathSizes

```
// Returns the number of boxes on all the surrounding
// 2-paths. (Some may be counted more than once.)
int CountSurroundingPathSizes(PtrToBoxRecord box) {

  int count = 0;
  FOR_EACH_OPEN_DIRECTION(box)
  {
    if (f_box->twopath)
      count += f_box->twopath->pathsize;
  } END_FOR_EACH_OPEN_DIRECTION;
  return count;
}   // CountSurroundingPathSizes
```

MinimumTwoPath

```
// Returns the smallest 2-path.
PtrToTwoPathRecord MinimumTwoPath () {

  ASSERT(!gList[3]);
```

Are You Prepared For An Emergency?

RADGAD.com SM

The Place for Useful Gifts & Gadgets. SM

www.radgad.com

PO Box 5200 Westlake Village, CA • 91359-5200 • Outside U.S. Canada: 805/ 494-9797 • Fax: 805/494-9798
Toll Free: 877-5-RADGAD (877-572-3423)

```
  if (! gPathsComputed)
    ComputePaths();

  int minvalue = INT_MAX;
  PtrToTwoPathRecord min, p;

  if (gTwoPaths[1])
    return gTwoPaths[1];

  if (gTwoPaths[2])
    return gTwoPaths[2];

  for (p = gTwoPaths[0]; p; p = p->next)
  {
    int size = p->pathsize;
    if (p->loop != LOOP)
    {
      size += 3; // Penalize 2 since only only get a 2-handout instead of a 4-
handout
      // Also give 1 penalty since we'd much rather be left with non-loop
      // than a loop at the very end, because we won't get the last handout.

      if (p->loop == ALMOST_LOOP && size < minvalue)
      {
        PtrToBoxRecord box = p->path;
        FOR_EACH_OPEN_DIRECTION(box)
        {
          if (NumberOfEdges(f_box) == 1)
          {
            // f_box is connected to both edges of this 2-path
            // plus one other 2-path. Filling in this 2-path
            // will cause the two 2-paths to be connected, so
            // the size should be calculated as the total of
            // both paths + 1 for the 1-box + 3 for the penalty
            // (size is subtracted because the almost loop is
            // counted twice)
            size = 4 + CountSurroundingPathSizes(f_box) -
size;
            break;
          }
        } END_FOR_EACH_OPEN_DIRECTION;
      }
    }

    if (size < minvalue)
    {
      minvalue = size;
      min = p;
      if (size == 3)
        break;
    }
  }

  return min;
}   // MinimumTwoPath
```

HandOut

```
// Given a HandOut type (either 2 or 4) and that every box on the 3's list is
// either a 2 or hand out, HandOut finds a hand out of the indicated type in
// the 3's list and sets move to do this hand out.
void HandOut (int ho_type, MoveRecordType &move) {

  PtrToBoxRecord temp;
  pathdirection pdir;

  FOR_LIST(temp, 3)
  {
    if (temp->twopath->pathsize == ho_type)
    {     // HandOut
      pdir = OutPathDirection(temp);
      move.box = temp->pnext[pdir];
      FOR_EACH_OPEN_DIRECTION(move.box)
      {
        if (!IsInsideBoard(f_box) || temp != f_box)
        {
          move.dir = f_dir;
          return;
        }
      } END_FOR_EACH_OPEN_DIRECTION;
      break;
    }   // if
  }
}   // HandOut
```

HardHeartedHandout

```
// Given a 2-box on a 2-path of length two, returns the
// correct direction that a new edge should be placed.
direction HardHeartedHandout (PtrToBoxRecord box) {

  FOR_EACH_OPEN_DIRECTION(box)
  {
    if (NumberOfEdges(f_box) == 2)
      return f_dir;
  } END_FOR_EACH_OPEN_DIRECTION;
  ASSERT(0); // should never get here
  return left;
}   // HardHeartedHandout
```

FindSafeMoveOnList

```
// Tries to find a safe move on gList[listindex]
// If so, set move and return true
// If not, these boxes are now unsafe. Return false
Boolean FindSafeMoveOnList(int listindex, MoveRecordType
&move) {

  PtrToBoxRecord box;

  // If any legal adjacent element is !=2 or outside board, take it
  FOR_LIST2(box, listindex, gListBookmark[listindex])
  {
    direction k, d;
    d = k = RandomDirection();
    do {
      if (!box->edges[d])
      {
        PtrToBoxRecord box2 = Go(box, d);
        if (!IsInsideBoard(box2) || NumberOfEdges(box2) !=
2)
        {
          gListBookmark[listindex] = box->next;
          move.box = box;
          move.dir = d;
          return true;
        }
      }
      d = NextDirection(d);
    } while (d != k);
  } END_FOR_LIST2(box, listindex, gListBookmark[listindex])

  gSafe[listindex] = false;
  return false;
}   // FindSafeMoveOnList
```

ComputerTurn

```
// Come up with a move for the computer
void ComputerTurn(MoveRecordType &move) {

  PtrToBoxRecord temp;

  // generate computer's move
  if (!gList[3])
  {
    if (!gList[0] && !gList[1])
    {     // must pick a 2-box from a 2-path
      PlayMoveOnPath(MinimumTwoPath(), move);
      return;
    }

    // (gList[0]!=NULL or gList[1]!=NULL) and !gList[3]

    // Try safe boxes on lists
    if (gSafe[0] && FindSafeMoveOnList(0, move))
    {
      return;
    }
    if (gSafe[1] && FindSafeMoveOnList(1, move))
    {
      return;
    }

    // All bad choices: all 0's and 1's are adjacent to 2's
    if (!gPathsComputed)
      ComputePaths();
```

```
  // now count how many 2's in a row— if one, take it
  // — if two, take it (put | between 2's)
  PlayMoveOnPath(MinimumTwoPath(), move);
  return;
}

// gList[3]!=NULL

if (gSafetyCheckNeeded)
{
  if (gSafe[0])
    CheckSafety(0);
  if (gSafe[1])
    CheckSafety(1);
  gSafetyCheckNeeded = false;
}

if (gSafe[0] || gSafe[1] || !gList[2] ||
                          gMaxScore - gTotalScore <= 4)
{
  // Note: If the number of untaken boxes is <= 4, then always
  // take a box, since hand-outs can't be better
  PlayAny(gList[3], move);
  return;
}

if (!gPathsComputed)
  ComputePaths();

// unsafe && gList[2]!=NULL && gList[3]!=NULL
// while not at end of 3's list do
//  if the box connected to the 3-box is not a 2-box (or is outside the board)
//     then take it
FOR_LIST(temp, 3)
{
  if (!temp->twopath)
  {
    PlayAny(temp, move);
    return;
  }
}

// unsafe & gList[3]!=NULL & all 3-boxes are part of 2-lists
// Take all 3's that will not ruin HandOut possibilities
FOR_LIST(temp, 3)
{
  // if connecting 2-path is not of length 2 or 4 then take it
  ASSERT(temp->twopath);
  int size = temp->twopath->pathsize;
  if (size != 2 && size != 4)
  {
    PlayAny(temp, move);
    return;
  }

  // if 2-path looks like 3-2-2-2 then take it
  if (size == 4)
  {
    pathdirection pdir = OutPathDirection(temp);
    PtrToBoxRecord box2 =
          temp->pnext[pdir]->pnext[pdir]->pnext[pdir];
    if (NumberOfEdges(box2) == 2)
    {
      PlayAny(temp, move);
      return;
    }
  }
}

// Now all 3-paths look like 3-2 or 3-2-2-3. These both are
// are hand out possibilities...
//
//        2-path                    |_   | -> |_ _|
//
//        4-path                    |_ _ _ _| -> |_ _|_ _|

  // Before we hand out though, let's see if there are any
  // tiny 2-paths where 2 or 4 handouts don't exist.
  // These change the initiative.
  if (!gList[1])
  {    // special unsafe 1-box configurations are not counted
    int tiny2paths = CountTinyTwoPaths ();
    if (ODD(tiny2paths))
    { // We do not need to retain the initiative because it will
      // change in our favor anyway
      PlayAny(gList[3], move);
      return;
    }
  }

  // It is possible to have more than one HandOut available so
  //  count the number of each kind of HandOut possibility.
  int num2ho, // the number of length 2 hand out possibilities
    num4ho;  // the number of length 4 hand out possibilities
  num2ho = num4ho = 0;
  FOR_LIST(temp, 3)
    if (temp->twopath->pathsize == 2)
      num2ho++;
    else
      num4ho++;
  num4ho /= 2; // were counted twice
  if (num2ho + num4ho > 1)
  { // More than one hand out available so grab another square first
    if (num4ho)
    {     // take the box from a length 4 2-path
      temp = gList[3];
      assert(temp->twopath);
      while (temp->twopath->pathsize != 4)
        temp = temp->next;
      PlayAny(temp, move);
      return;
    }

    // more than one length=2 hand outs available so play one
    PlayAny(gList[3], move);
    return;
  }

  // only one handout possibility

  // Now see if it's worth it, or we should just be greedy.
  if (ExpectedScore(gList[3]->twopath) * 2 <
                              gMaxScore - gTotalScore)
  { // It's not worth it: just be greedy and don't give handout
    PlayAny(gList[3], move);
    return;
  }

  if (num2ho)
  {    // do a TPH
    HandOut(2, move);
    return;
  }

  ASSERT(num4ho == 1);
  // do a FPH (ickk)
  HandOut(4, move);
}  // ComputerTurn
```

MakeRealMove

```
// Makes a move on the board. Returns true if a new block was formed.
Boolean MakeRealMove(MoveRecordType move, who whoseturn) {

  //Take care of the player's move
#ifdef DRAWING
  DrawMove(GetX(move.box), GetY(move.box), move.dir);
#endif
  int oldscore = gScore[whoseturn];
        // score of the current player before his move
  Boolean checkneeded = false;

  AddEdge(move.box, move.dir, whoseturn);

  PtrToBoxRecord box2 = Go(move.box, move.dir);
  AddEdge(box2, OppositeDirection(move.dir), whoseturn);
```

```
  if (gPathsComputed)
  {
    Boolean same2path = box2->twopath &&
        (move.box->twopath == box2->twopath) &&
        NumberOfEdges(move.box) == 3;
    if (same2path)
      box2->twopath = NULL;
          // See if updating move.box's 2-path updates this too
    UpdatePathFrom(move.box);

    // No need to update the path again if the boxes are still on the same 2-path

    if (!same2path || !box2->twopath)
      // Note that we want box2's 2-path to remain NULL if set above,
      // since that will cause a new 2-path to be created
      UpdatePathFrom(box2);
  }

  return oldscore != gScore[whoseturn]; // did score change?
}   // MakeRealMove
```

CHALLENGE INTERFACE

```
// Converts from input data types to engine data types. Modifies box,d
void ConvertFromDotLine(MoveRecordType &move, const DotLine
&dotline) {

  int x = dotline.dot1.col + 1;
  int y = dotline.dot1.row + 1;

  if (x > gBoardSizeX)
  {
    -x;
    move.dir = right;
  }
  else if (y > gBoardSizeX)
  {
    -y;
    move.dir = down;
  }
  else if (dotline.dot2.col > dotline.dot1.col)
    move.dir = up;
  else // (dotline.dot2.row > dotline.dot1.row)
    move.dir = left;

  move.box = BOX(x, y);
}

// Converts from engine data types to output data types,    Modifies dotline
void ConvertToDotLine(MoveRecordType move, DotLine &dotline)
{

  int x = GetX(move.box);
  int y = GetY(move.box);
  direction d = move.dir;

  if (d == right)
  {
    d = left;
    ++x;
  }
  else if (d == down)
  {
    //d = up; // not strictly necessary :-)
    ++y;
  }
  dotline.dot1.col = x - 1;
  dotline.dot1.row = y - 1;
  dotline.dot2 = dotline.dot1;
  if (d == left)
    dotline.dot2.row++;
  else // d == up
    dotline.dot2.col++;
}
```

InitDots

```
/* Play begins with a call to your InitDots routine, where you are
given the size of the game board (boardSize), an indicator of who
plays first (playFirst), and a pointer to a CWindow (passed as a
WindowPtr because that's what most toolbox routines expect). In
that window, you will be required to display the progress of the
game as it proceeds. */
void InitDots(
  short boardSize,      // number of dots per row/col in board
  Boolean /* playFirst */,   // true if you play first, false of opponent plays first
  WindowPtr dotWindow // color window where you should draw game results
) {
#pragma unused(dotWindow)

  gCurrentPlayer = firstplayer;  // whose turn it is to play
  gBoardSizeX = gBoardSizeY = boardSize - 1;
  Initialize();
  InitialDrawing();
}
```

OpponentMove

```
/* After your opponent has played, your OpponentMove routine
will be called one or more times, once for each move made by
your opponent. The move will be provided in the opponentLine
parameter, for use in display and in updating your data structures. */
/* After each of your moves, and after notification of each
opponent move, you should display the move and the updated
game state in the dotWindow. The window should also display
the number of squares completed by each player. The details
of the display are left to you, as long as the display is correct. */
void OpponentMove(
  const DotLine opponentLine
  // line formed by your opponent on previous move
) {

  MoveRecordType move;
  ConvertFromDotLine(move, opponentLine);
  if (!MakeRealMove(move, gCurrentPlayer) &&
                        gTotalScore != gMaxScore)
    NextPlayer(gCurrentPlayer);
}
```

PlayDots

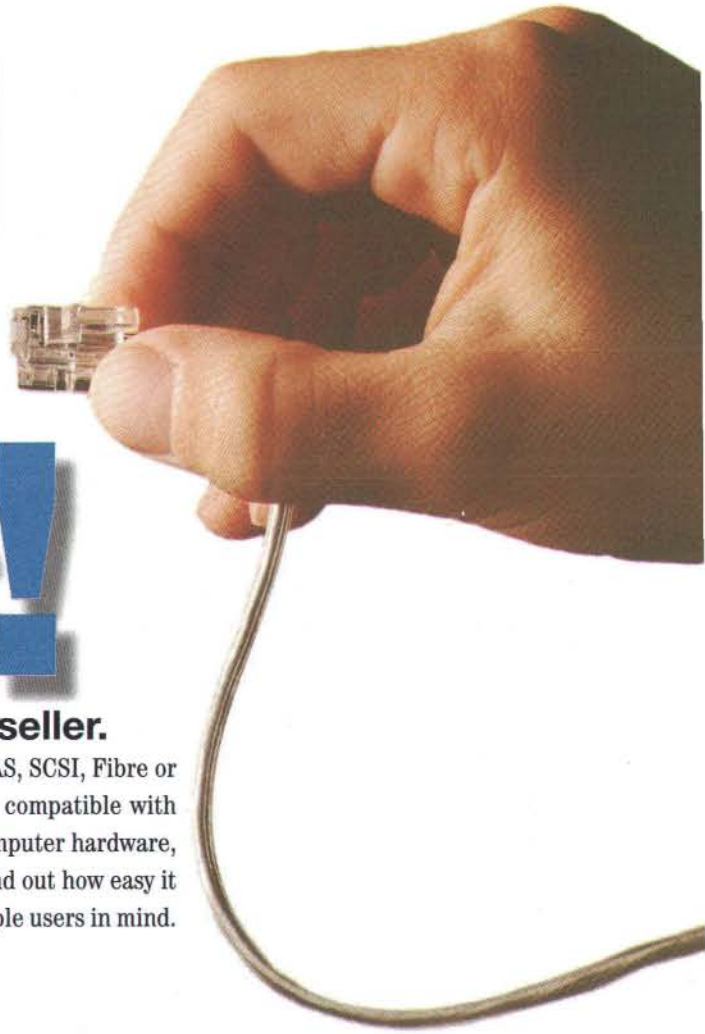
```
/* When it is your turn to move, your PlayDots routine will be
called. Your code should select the most advantageous move and
return it in yourLines[0]. If that move forms a square, you can
select an additional move, store it in yourLines[1], and continue
as long as squares are formed. PlayDots should return the number
of moves you made during your turn. */
short /* number of lines generated */ PlayDots(
  DotLine yourLines[]   // return the lines you form here
) {

  Boolean madeBox;
  MoveRecordType move;
  int moves = 0;
  do {
    ComputerTurn(move);
    madeBox = MakeRealMove(move, gCurrentPlayer);
    ConvertToDotLine(move, yourLines[moves]);
    ++moves;
  } while (madeBox && (gTotalScore != gMaxScore));

  if (gTotalScore != gMaxScore)
    NextPlayer(gCurrentPlayer);
  return moves;
}
```

TermDots

```
/* When all of the squares have been formed, your TermDots routine
will be called. You should deallocate any dynamically allocated
memory and perform any other cleanup required. */
void TermDots() { // return any storage you allocated

  delete [] gBoxes;

  for (int i=0; i<3; ++i)
  {
    PtrToTwoPathRecord p = gTwoPaths[i];
    while (p)
    {
      PtrToTwoPathRecord p2 = p->next;
      delete p;
      p = p2;
    }
  }
}
```

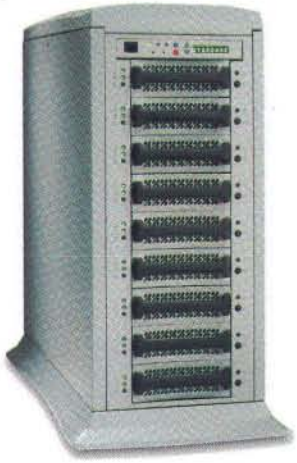
Plug into us!

Mac



CompuPlus, Inc., the Apple experts reseller.

Get complete solutions for all your storage needs whether it's SAN, NAS, SCSI, Fibre or Firewire from CompuPlus, Inc. Besides Apple, our products are also compatible with UNIX, Windows NT and Linux. At CompuPlus, Inc. we are all about computer hardware, so call us today at **(800) 536-7378** and talk to one of our experts to find out how easy it is to plug into this feature packed storage solution, integrated with Apple users in mind.



StorCase InfoStation™

Ready for the future when you are!

Simply attach the field-upgradable InfoStation 9-bay storage enclosure into your Apple G4 USB port and experience up to 1.6 TeraBytes of plug-and-play, worry-free storage for RAID or JBOD applications. Get real-time updates of chassis environmentals via InfoStation's on-board monitoring utility or add the optional SNMP upgrade module, which will be shipping in August, along with OpenView software, and check the health of your storage system via the internet anywhere in the world. Or, opt to be E-mailed or paged about any chassis environmental changes. As with most StorCase products, the InfoStation includes an industry-leading 7-year warranty and FREE 24/7 technical support.

StorCase® TECHNOLOGY
A Kingston Technology Company

Seagate Baracuda™

180GB hard drive

Seagate provides the industry's leading performance disk drives. You can find them used in some of the most demanding applications: internet servers, video editing workstations, network file servers and enterprise servers.

Seagate

To order, contact your sales rep today at CompuPlus, Inc.

(800) 536-7378

www.compuplusinc.com

CompuPlus Inc.
the system integrators

130 McCormick Ave., Ste. 106, Costa Mesa, CA 92626
E-mail: sales@compuplusinc.com

```
// Screen.cpp
// Copyright 2001 Jeff Mallett. All rights reserved.


#include <stdio.h>
#include <string.h>

#include <QuickDraw.h>
```

DRAWING CONSTANTS

```
// The full board can't actually fit on the window. I get less than 20x20
// boxes showing in the test code window. Therefore only draw
// the upper-left portion of grid if it's that big.
// The maximum dimensions to draw can be adjusted here if the window
// size is increased:
const int MAX_X_DRAW = 25;
const int MAX_Y_DRAW = 25;

const int BOX_WIDTH = 15;       /*pixel distance between adjacent dots*/
const int SCORE_TITLE_Y = 20;
const int SCORE_Y = 38;         /*distance of scores from top of screen*/
const int GRID_Y = 50;
const int SCORE_X[2] = {40, 130};
const char PLAYER_INITIAL[2] = { '1', '2' };
        /*PLAYER_INITIAL[w] is the initial of player w*/

enum direction {left=0, up, right, down};
enum who {firstplayer=0, secondplayer};

extern int gBoardSizeX;
extern int gBoardSizeY;
```

PROTOTYPES

```
void ScreenLoc (int x, int y, int& i, int& j);
void DrawEdge (int x1, int y1, int x2, int y2);
void DrawDots();
void DrawScore (who w);
void DrawInitial (int x, int y, who person);
void DrawMove(int x, int y, direction d);
void DrawScoreTitle(who person);
void InitialDrawing();
```

DRAWING PROCEDURES

```
/*Given box coordinates x,y returns coords of top-left point of box (i,j)*/
void ScreenLoc (int x, int y, int& i, int& j) {

  i = x * BOX_WIDTH;
  j = y * BOX_WIDTH + GRID_Y;
}   /*ScreenLoc*/

/*Given the coordinates of two boxes x1,y1 and x2,y2, this will draw in*/
/* an edge connecting the dot in the upper left hand corner of the two boxes.*/
void DrawEdge (int x1, int y1, int x2, int y2) {

  int i1, j1, i2, j2;
  ScreenLoc(x1, y1, i1, j1);
  ScreenLoc(x2, y2, i2, j2);
  MoveTo(i1, j1);
  LineTo(i2, j2);
}   /*DrawEdge*/

/*Print the dots on the screen which form the playing board*/
void DrawDots() {

  int hor, ver, i, j;
  Rect rect;
  int maxh = gBoardSizeX;
  int maxv = gBoardSizeY;

  if (maxh > MAX_X_DRAW)
    maxh = MAX_X_DRAW;
  if (maxv > MAX_Y_DRAW)
    maxv = MAX_Y_DRAW;
  ++maxh;
  ++maxv;

  for( hor = 1; hor <= maxh; hor++)
    for( ver = 1; ver <= maxv; ver++) {
      ScreenLoc(hor, ver, i, j);
      SetRect(&rect, i, j, i + 1, j + 1);
      PaintOval(&rect);
    }
}   /*DrawDots*/

/*Update a player's score on the drawing window*/
void DrawScore (who w) {

  extern int gScore[];
  int n;
  Rect rect;

  n = SCORE_X[w];
  SetRect(&rect, n, SCORE_Y - 11, n + 40, SCORE_Y + 2);
  EraseRect(&rect);

  char s[256];
  sprintf(s, "%ld", gScore[w]);

  MoveTo(n, SCORE_Y);
  DrawText(s, 0, strlen(s));
}   /*DrawScore*/

void DrawInitial (int x, int y, who person) {

  if (x <= MAX_X_DRAW && y <= MAX_Y_DRAW)
  {
    int hor, ver;

    ScreenLoc(x, y, hor, ver);
    hor += (BOX_WIDTH / 2) - 3;

    ver += (BOX_WIDTH / 2) + 6;
    //TextFace(bold);
    MoveTo(hor, ver);
    DrawText(&PLAYER_INITIAL[person], 0, 1);
    //TextFace(normal);
  }
}   /*DrawInitial*/


void DrawMove(int x, int y, direction d) {

  if (x <= MAX_X_DRAW && y <= MAX_Y_DRAW)
  {
    switch (d) {
    case left :
      DrawEdge(x, y, x, y + 1);
      break;
    case up :
      DrawEdge(x, y, x + 1, y);
      break;
    case right :
      DrawEdge(x + 1, y, x + 1, y + 1);
      break;
    case down :
      DrawEdge(x, y + 1, x + 1, y + 1);
      break;
    }
  }
}

void DrawScoreTitle(who person) {

  Rect rect;
  int n = SCORE_X[person];
  SetRect(&rect, n, SCORE_TITLE_Y - 11, n + 40,
                                  SCORE_TITLE_Y + 2);
  EraseRect(&rect);

  MoveTo(n, SCORE_TITLE_Y);
  DrawText(&PLAYER_INITIAL[person], 0, 1);
}

void InitialDrawing() {

  DrawScoreTitle(firstplayer);
  DrawScoreTitle(secondplayer);

  int x = SCORE_X[firstplayer]-10;
  int y = SCORE_TITLE_Y+4;
  int x2 = SCORE_X[secondplayer]+20;
  MoveTo(x, y);
  LineTo(x2, y);
  MoveTo((x+x2)/2, SCORE_TITLE_Y-15);
  LineTo((x+x2)/2, SCORE_Y+15);

  DrawDots();
}
```

Not Your Average Wireless Access Point

NetLINE Wireless Broadband Gateway

BEST of SHOW
MacWEEK.com presents
Macworld SF 2001
AWARD

Farallon
www.farallon.com

Share your **high-speed** Internet connection with both **wired** and **wireless computers** while protecting your data from unauthorized access.

The NetLINE Wireless Broadband Gateway allows you to share a high-speed Internet connection, such as Cable or DSL, with multiple computers using just a single IP address from your ISP. Support for both 802.11b wireless and standard Ethernet connections are built-in to this router and access point all in one.

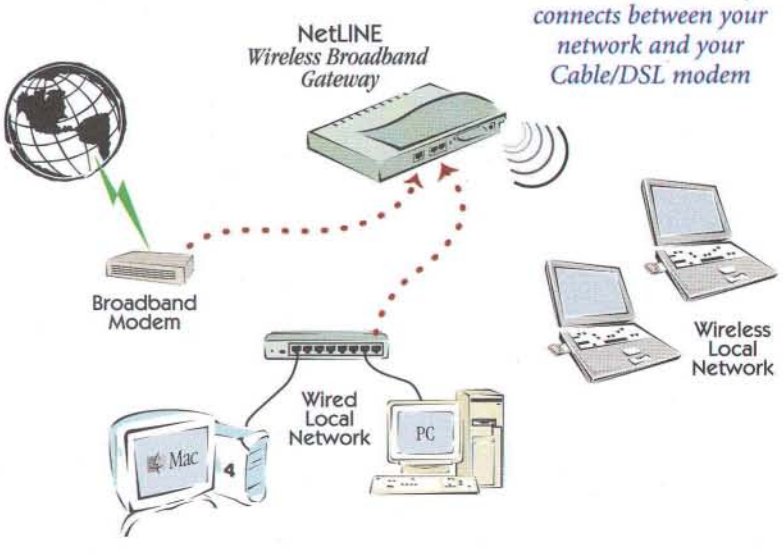
The built-in firewall will protect your important files by preventing unauthorized access via the Internet and the web-based configuration makes set-up a breeze.

100% compatible with SkyLINE, Airport, and all other 802.11b wireless cards.

For more information contact Dr. Farallon at 1-800-613-4954 or visit us on the web at **www.farallon.com.**

Available @ the DEPOT.

The NetLINE Wireless Broadband Gateway connects between your network and your Cable/DSL modem



By John C. Welch

A Network Geek in the Big Apple

The Expo Done Gone

Well, the expo is over, and another week of mayhem is in the past. While the general news reports make this out to be a disappointing Expo, my experience was the exact opposite. I found this to be one of the first expos where I could easily find products for my world. Considering my world is network management and administration, and of late, training, MacWorld Expo can be something of a disappointment for me, even when the rest of the Mac community is excited. In fact, I had my first indication that this was going to be a 'most excellent' IT MacWorld before the show had started.

On July 12th, Dartware released their port of net-snmp 4.2.1 for Mac OS X. This happened with very little fanfare, yet for Mac OS X, and especially Mac OS X server, this is of critical importance. SNMP, which stands for Simple Network Management Protocol is a cross-platform standard for managing networks. It allows for remote status checking, and via the trap mechanism, some ability to perform management actions on remote machines. Virtually every computing platform supports SNMP, and although it has its problems, and detractors, it gets the job done, for the most part. Unfortunately, up until the Dartware release, there was no SNMP support in Mac OS X. This was, in my opinion, and I think I speak for many other managers here, a glaring omission. It meant that even on a Unix network, Mac OS X would require special tools, or be unable to use many of the standard management tools out there, such as CA Unicenter, HP OpenView, etc. This was a major obstruction to Mac OS X being a first class network citizen.

But thanks to the folks at Dartware, we have a first – class SNMP implementation for Mac OS X. Even better, it's an open source implementation. Better still, it's based on the BSD license, and not the GPL. This last difference is important, especially to Apple. For a corporation, such as Apple, the GPL is a minefield. The GPL requires you to post any source code for any product you create that incorporates GPL – licensed source code. So for Apple, if they use GPL code in Aqua, for example, they would be required by law to post all the source code that relates to the parts of Aqua using GPL code. In this sense, the GPL is almost a kind of viral license. Now, the argument could be made that the FSF, (Free Software Foundation), who owns the GPL copyright doesn't really tend to sue people over this. But legally, that isn't the point. The fact is, if the FSF *did* choose to take this action, then Apple would be forced to comply, or withdraw immediately the affected components until the GPL code could be removed. Considering that the head of the FSF, Richard Stallman, is well known for his intense dislike for Apple as a corporate entity, betting on his good will is not a wise choice.

The BSD license, on the other hand, makes no such requirement. You are allowed to use the code under the license freely, and although you are encouraged to return the code of the results to the community, you don't have to. This is a far more 'open' open source/free software license than the GPL, and for Apple, a far better one. This means that at some point, if they desired, they could incorporate net-snmp into Mac OS X and Mac OS X Server, and not have to worry about what corporate information they would be required to release. It means that the chances of this happening are far higher than they would otherwise be, and that we, as the community of Mac network managers have a better chance of getting a much needed improvement faster than it would have happened otherwise.

The Keynote

You cannot discuss MacWorld Expo without mentioning the Keynote. Much has been made of this already, and I think that many of the opinions show signs of being made quickly, without any real thought. My take on it was that this was a maintenance keynote, and deliberately so. Some conversations I had seem to prove this out. In all honesty, there wasn't a lot of "wow" this time around. Yes, the new G4s are nice, and the new speaker system in them is nice, and people seem to like the new cases a lot. Faster iMacs are always good, and though I personally liked the Flower Power design, enough folks disagreed with me, so the new cases are conservative in color. We never actually got to see the 700MHz iMacs, so there is some room there for Apple to introduce something radical at the high end. No new iBooks, no big surprise, they aren't that old. No new Titaniums, no CD-RW options for them. Again, no big surprise on that score, I have yet to hear about a CD-RW mechanism that is thin enough to fit in a TiBook. The announcements from the '10 on X' vendors were interesting enough, especially the way Adobe seems to have re-invented Publish and Subscribe. The only sour note was Quark, who, in spite of being ridiculously late with Xpress 5, insists on telling us that the Carbon version of Xpress is right on schedule. Well, in geological terms, I suppose it is. Tip from a customer, if you are late on a product, don't tell me how cool the next version will be. It makes you look silly.

The scandal over the pitched camera was silly. First of all, how much thought does it take to ensure the thing is plugged in and working before the keynote. Secondly, Steve didn't wing, fling, or pitch the thing at anyone. He tossed it to someone about fifty feet away. Physics tells us that to get the camera to fly fifty feet, it needs a certain amount of force. The person on the receiving end bobbled the camera and the batteries got dropped. That's it, no attempt by Steve to take someone's head off with it.

The Mac OS X 10.1 demo was a good idea, it was necessary to see that we will be getting the OS that OS X is supposed to be. Like

John Welch <jwelch@macseminars.com> is a Training and Unix Specialist for Complete Mac Seminars, the premiere Mac OS training organization. He has over fifteen years of experience at making computers work. His specialties are figuring out ways to make the Mac do what nobody thinks it can, showing people that the Mac is the superior administrative platform, and teaching them how to use it as just that.

many, I would have liked a CD to have been taped to my seat, but I can wait until September. The iDVD 2 demo went on *way* too long, as did the commercials. Then again, the seats in the keynote room weren't exactly comfortable for long periods of sitting.

From my point of view, the most telling moment was Steve thanking the families of Apple employees. As soon as I saw that, the keynote really made sense. This was Steve essentially apologizing to the partners and families of the people under him for the insane work hours they have been putting in. Think about that, and think about the 2001 time line that we saw at the keynote. With the exception of the iMac, Apple has, in the last year, come out with a new hardware lineup. Even the G4 tower has a new motherboard design, along with the new case. Mac OS X has been released, and had four updates. iMovie, AppleWorks, FileMaker Pro, FileMaker Pro Server, and iTunes are all native. Apple has participated in, or put on, three trade shows, a developer conference, and sent people to MacHack. Apple has been killing itself, and these people need a break. A chance to go back to twelve hour days instead of twenty hour days. A chance to send people home on the weekends. A chance to get eight hours of sleep instead of two. I know some of the Apple folks, and they look *exhausted*. Remember, Apple isn't a startup. Even under the best of economic times, this company is not going to have a 200% increase in stock value. These people have salaries, they can leave Apple, and work elsewhere. They can also get so burnt out that they leave the business entirely. That would be a shame, as these are also some of the smartest, most creative, and at least for the folks I know, some of the nicest people I have ever met.

So, at least to me, that was what the keynote was about. Some new stuff, a reminder of exactly how furiously Apple has been working since January, and a chance to give Apple employees a day off or two, maybe even a vacation.

The Show

The show floor and seminars were packed with people, even if the vendor numbers were down this time. From an IT point of view, Mac OS X has had a huge impact on the show, as compared to San Francisco's show. The amount of products available for the new operating system is simply huge, and even allowing for most of it being pre-release or other forms of beta, I was more than impressed.

On the traditional networking front, Netopia had not only their Timbuktu pre-release out for demonstration, but an early alpha of netOctopus, their network administration framework as well. Timbuktu for Mac OS X allows you to control not only other Mac OS X machines, but Wintel machines, and Mac OS 9 or earlier Macs too. The implementation for Mac OS X is as full – featured as the other versions, including the AppleScript dictionary. Other features include live dock displays of remote sessions, and a floating tile that replaces the Mac OS 9 menu extension. The tile is patterned the same as the Mac OS X menu bar, so if you drag it to a clear spot on the menu bar, it looks like a menu extension. Hopefully, with version 10.1 of Mac OS X, this can actually live in the menu bar, although I would like to see the menu tile become an optional convenience. The netOctopus alpha I saw was looking really good, and has some new features that users of this excellent product have wanted for a while, including the ability to get the machine hardware number of the Mac via the Apple System Profiler. Color information is now included, which, thanks to Apple identifying machines via color, is a needed informational item. The SNMP module was there, and hopefully, this new version will add support for SNMP traps, bringing netOctopus up to the level of higher end network management applications. For an application which is still the only cross-platform management program that can run on a Mac as both server and client, it is good to see that Netopia's commitment to the product is unwavering.

Another company with a similar unique position in the Mac market is Dantz, and to show their continued drive to make Retrospect the best backup program on the planet, much less the Mac market, they announced the first OS X – native version of Retrospect Server, version 5.0. This is a Carbonized version of Retrospect Backup Server, and is more or less a port of the 4.3 Mac server. As such, other than some speed improvements, and the ability to handle Mac OS X clients properly, there are not many architectural improvements to the server. However, Dantz is working on bringing Retrospect into the Cocoa world, and working on bringing over the improvements they made when they created the new server architecture for their Windows products.

This will allow Dantz to give Mac administrators a high end backup server that can handle larger networks, and higher end backup devices better than the current versions. On the client side, Dantz has done an excellent job with the Mac OS X and Mac OS X Server betas of their clients, both Cocoa front ends to background daemons. This split architecture, common in the Unix world has allowed for some pretty significant performance increases. My own experiences have shown a consistent two to three times increase in backup speed between the Mac OS 9 and Mac OS X clients. (On one machine, a G4/450, on a clean switched 100 Mb network, the backup speed went from 60 – 90MB/min. to 120 – 230MB/min, with no other changes than OS and client.) The other performance increase is that other than an increase in disk activity, the user doesn't notice the backup happening, so they can avoid having to set the client performance slider to the low end of the performance scale, avoiding the slowdown in backup speeds that always results. Dantz is also seeking feedback on client support for other Unix operating systems, so if you want to see a specific Unix supported, let them know.

FileMaker announced the server version of their database application, a Cocoa version. Like the Retrospect client, FileMaker Pro Server features a factored interface. This allows the database server application to run independently of the user interface on the server. This allows the server to run without an active login, and without the overhead of the user interface. While new for Mac users, this is how databases such as Oracle and DB2 run, and moving to this split architecture is critical for FileMaker to begin giving FileMaker Pro Server the kind of performance and capability that it needs to keep on growing as a database server. Another advantage to this split implementation is that it makes porting FileMaker Pro Server to other Unix architectures easier, and FileMaker has done exactly that, by also announcing the release of a Red Hat Linux version of FileMaker Pro Server. This now allows FileMaker administrators to run their servers on higher-end hardware, while still maintaining a Unix server environment, avoiding the issues created by introducing Windows servers into such an environment.

Mac server stalwart WebSTAR is also being revamped for Mac OS X, even in the face of the major question of why? After all, if you get

Apache free with Mac OS X, why pay money for a third party web server? Well, WebSTAR is more than just a different version of httpd, the web daemon in Mac OS X. First off, WebSTAR has been completely rewritten for Mac OS X. Although 4D started to try and just do a Carbon port of the Mac OS 9 version of WebSTAR, the complexities in the old WebSTAR code created an immense barrier to doing this. In the end, it became a better, idea to rewrite the server from the ground up. This allowed for several advantages. First of all, the WebSTAR server process is actually a BSD Unix application, not really Cocoa or Carbon. This allowed 4D to avoid any overhead for an administrative interface within the server, adding speed and stability. The administrative interface is a Java application, allowing it to be run from not only Mac OS X, but also from Windows, Solaris, or any platform with adequate Java support. 4D was also able to add Altivec optimizations where appropriate, so not only can WebSTAR take advantage of multiple processors, but also takes advantage of the vector units on those processors.

WebSTAR has some other differences from Apache that give it advantages over Apache. One of them is the security model that WebSTAR uses. WebSTAR has its own user database, rather than using the NetInfo database in Mac OS X. While inconvenient in the sense that an admin has to set up users in two places, this means that if someone is able to crack an admin password in WebSTAR, they do not automatically have the keys to the kingdom. They may be able to do evil things to WebSTAR, but they still have to get a different password to root the OS X box WebSTAR is running on. WebSTAR also doesn't run as root, so even if a cracker manages to get WebSTAR to run unauthorized code, they are not doing this as root, limiting the damage they can do easily. WebSTAR does not run as part of inetd, so trying to use WebSTAR as a way to crack this root level service fails as well. WebSTAR also includes support for AppleEvent CGIs, so WebSTAR administrators with a library of AppleScript CGIs don't have to convert them to Perl, or some other language. Finally, 4D is conducting an extensive line – by – line review of WebSTAR's source code, so that any code- level bugs that could allow security breaches can be caught. 4D is very aware of the expectations that WebSTAR's success in repelling crackers has created, and are sparing no effort to live up to those expectations.

Another application that has long been the only one of its type on the Mac is 4-Site Fax server. The only multi-line, LAN – friendly fax server on the Mac, 4-Site has languished through some changes of ownership, and its user base wondered when it would simply be killed off. Well, luckily, some former developers and users of 4-Site have bought it, and were showing off the OS X version. Version 5 of 4-Site is a Mac OS X – native server, with some excellent new capabilities. First off, it now supports TCP/IP, so the former requirements the Mac server had for AppleTalk are gone. Also, the new version features client - independent email integration, so users can send and receive faxes via email, without needing a fax client. This integration has an added benefit in that you can set up a machine to relay emails to the fax server, so that you could allow for multiple users per client, a lightweight way of taking care of the fax needs of those users who don't need a full-featured fax client. When using the email integration, faxes are received as PDF documents, so things like OCR and other post – processing operations are simplified. The client itself is a Java application, and when I asked if this meant that any platform supporting Java, outside of Windows and Mac OS X could use it, the reply was, "I don't see why not. We haven't tested it, but it should work." If the 4-Site client is able to achieve this level of cross-platform support, this would make it one of the leading fax servers on the overall market, not just the Mac market. The new server, a Carbon application, will be able to handle up to 16 in/ outbound lines, giving 4-Site a serious fax handling ability to start with. If 4-Site is able to work with the major telephony hardware developers, then Mac OS X could easily become a major force in that market.

Our final entry is Dartware LLC, and the release of a Mac OS X – native version of its InterMapper application. This is one of the best network monitoring applications available on any platform and Mac OS X gives it the stable base that it needs to run 24x7. InterMapper not only is able to use SNMP to monitor a network, but can use other protocols such as POP3, IMAP, HTTP, HTTPS, FTP, SMTP, FileMaker Pro Server, etc. to make sure that not only is the machine up on the network, but that the services it provides to the network are functioning as well. InterMapper takes a different approach to this than many similar applications. Instead of a client agent – based approach, which checks these services internally on the machine they run on, or via a loopback mechanism, InterMapper uses external queries, of the same type that a client for that service would use. So it sends out POP3, FTP, SMTP queries, and waits for the correct response. This is important for two reasons. First of all, especially on Unix servers, the server itself can still be running, even thought the services it is responsible for have crashed. So simply seeing if the box is running is not enough. Secondly, you can have a case where the server has gone deaf, and isn't processing external requests, even though it shows the services as running. By checking these services as a client would, the status of the service can be more accurately checked. The SNMP capabilities of InterMapper are top notch as well. Trap notifications to InterMapper are supported, and InterMapper can send its own SNMP traps to other applications, handy if InterMapper is monitoring part of a larger network run by things like HP OpenView, or Tivoli. The SNMP monitoring not only checks uptime, run status, packet errors, and other things, but is also used to create live traffic level maps, so you can look at an InterMapper map, and see traffic use in real time. (Hence the importance of Dartware's SNMP client release. This is invaluable for a Network Operations Center, (NOC). The alarms and warnings in InterMapper are configurable to your needs, and in addition to the trap report mechanism, InterMapper can notify you of problems via email, pager, and audible alarms.

Off the show floor, there was the customary excellent array of sessions and workshops available to educate attendees on almost every aspect of the Mac and the Mac OS in almost any implementation. While a conflict of interest prevents me from actually reviewing sessions, (I gave two sessions, and assisted on a third, also giving a pre-show workshop on Mac OS X.) I think that if you are going to take the time to go to an Expo, attending a session or two will greatly increase the value you get from the Expo.

Conclusion

So, in spite of a keynote without some amazing new announcement, the 2001 MacWorld Expo New York was an excellent show for network administrators. I will not begin to claim that I saw everything, much less covered it here, but hopefully, the items I was able to cover will give you a reason or impetus to attend your first expo, or to continue going.

ADVERTISER/ PRODUCT LIST

List of Advertisers

4D, Inc. 1
A.D. Software 31
AEC Software 37
Aladdin Knowledge Systems, Inc. 83
Aladdin Systems, Inc. 9
Appgen Business Software, Inc. 49
Apple Computer, Inc. 73
Applied Science Software 18
Art & Logic, Inc. 80
Ascending Technologies 20
Atimi Software Inc. 79
Automated Solutions Group 66
Blue World Communications, Inc 25
Bowers Development 69
Brad Sniderman 63
CompuPlusInc.com 89
Developer Depot 28-29
Dr. Bott LLC 21
edition.net 26
FairCom Corporation 5
Farallon Communications, Inc. 91
Felt Tip Software 65
Fetch Softworks 7
Full Spectrum Software, Inc. 19
garage.com 95
IOGEAR 17
Jiiva, Inc. 53
Lextek International 23
MacTech Magazine 67
MacTech Magazine 71
Marathon Computers, Inc. 43
Mathemaesthetics, Inc. 51
MCE (Mac Components Engineered) 57
Metrowerks, Inc. ..... BC
MindVision Software 60-61
Netopia, Inc. 11
O'Reilly & Associates, Inc. 55
Omni Group Development Inc 77
Onyx Technology 12
OpenBase International, Ltd. 35
Paradigma Software 23
Parallel Software 78
Power On Software 15
PrimeBase (SNAP Innovation) 39
Prosoft Engineering, Inc. 74-75
Quest Software, Inc. 45
RadGad 85
REAL Software, Inc. 33
Scientific Placement, Inc. 27
StoneTablet Publishing 31
SuSE Inc. ..... IBC
TeraGlobal Communications Corp. ..... IFC
Thursby Software Systems, Inc. 13
UNI SOFTWARE PLUS GMBH 47
Vanteon Corporation 76

List of Products

4D Web Edition * 4D, Inc. 1
Accounting Products * Appgen Business Software, Inc. 49
AppMaker * Bowers Development 69
AreaList Pro * Automated Solutions Group 66
c-tree Plus * FairCom Corporation 5
CalendarMonster * Ascending Technologies 20
Career Opportunities * Scientific Placement, Inc. 27
CodeWarrior * Metrowerks, Inc. ..... BC
DAVE * Thursby Software Systems, Inc. 13
Development & Testing * Full Spectrum Software, Inc. 19
DVIator * Dr. Bott LLC 21
eSellerate and VISE * MindVision Software 60-61
FastTrack * AEC Software 37
Fetch * Fetch Softworks 7
FunnelWeb * Quest Software, Inc. 45
HASP4 for Mac * Aladdin Knowledge Systems, Inc. 83
InstallerMaker * Aladdin Systems, Inc. 9
KVM * FireWire * USB * IOGEAR 17
Lasso * Blue World Communications, Inc 25
Law Offices * Brad Sniderman 63
Mac OS X Books * O'Reilly & Associates, Inc. 55
Mac OS X Porting Services * Art & Logic, Inc. 80
Mac OS X Porting Services * Atimi Software Inc. 79
Mac OS X Porting Services * Omni Group Development Inc 77
Mac OS X Porting Services * Parallel Software 78
Mac OS X Porting Services * Prosoft Engineering, Inc. 74-75
Mac OS X Porting Services * Vanteon Corporation 76
Mac OS X Porting Showcase * Apple Computer, Inc. 73
MacTech CD * MacTech Magazine 71
MacTech Magazine Subscription * MacTech Magazine 67
NetLINE Wireless * Farallon Communications, Inc. 91
Now Up-to-Date & Contact * Power On Software 15
Onix, RouteX, Brevity * Lextek International 23
OOFILE * A.D. Software 31
OpenBase SQL * OpenBase International, Ltd. 35
Porting & Development Services * Applied Science Software 18
PowerBook Essentials * Developer Depot 28-29
PowerBook Products * MCE (Mac Components Engineered) 57
PrimeBase * PrimeBase (SNAP Innovation) 39
Rackmount Solutions * Marathon Computers, Inc. 43
RapidMQ * Jiiva, Inc. 53
REALbasic * REAL Software, Inc. 33
Reseller Hosting * edition.net 26
Resorcerer * Mathemaesthetics, Inc. 51
Sound Studio * Felt Tip Software 65
Spotlight * Onyx Technology 12
Start ups * garage.com 95
StoneTable * StoneTablet Publishing 31
StorCase * CompuPlusInc.com 89
SuSE Linux * SuSE Inc. ..... IBC
TeraMedia * TeraGlobal Communications Corp. ..... IFC
Timbuktu Pro & netOctopus * Netopia, Inc. 11
Useful Gifts & Gadgets * RadGad 85
Valentina * Paradigma Software 23
VOODOO * UNI SOFTWARE PLUS GMBH 47

We know what's really on your mind.

Introducing CodeWarrior for Mac OS Version 7.0.

You want to make the move to Mac® OS X—and you still want to serve your Classic Mac customers—but you don't have the time to write multiple applications? Relax. CodeWarrior for Mac OS supports both platforms. In fact, CodeWarrior for Mac OS Version 7.0 lets you create a SINGLE application which runs on Mac OS 8.6, 9.x AND Mac OS X.

- Develop for targets: Classic Mac OS, Carbon, native Mac OS X, and Java
- Develop on Mac OS 8.6, 9.x and Mac OS X platforms
- Build native Mach-O OS X apps
- PowerPlant C++ framework
- Cross-platform development

Built for Mac OS X

So if you're making the move to Mac OS X, sink your teeth into the industry's leading software—CodeWarrior for Mac OS. For more information, or to order, visit www.metrowerks.com.

CodeWarrior®

Mac OS Development Tools

metrowerks®
Software Starts Here